Final Design Project

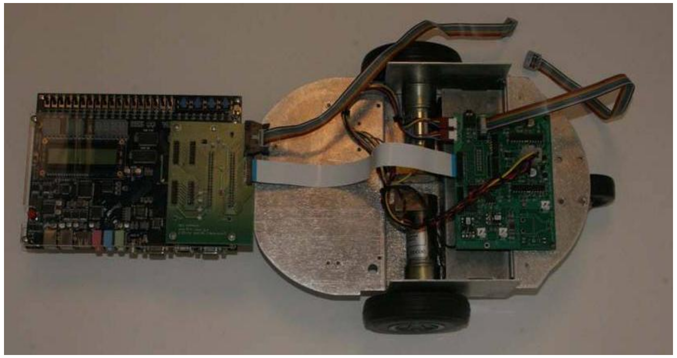
ECE2031 Spring 2013

Final project

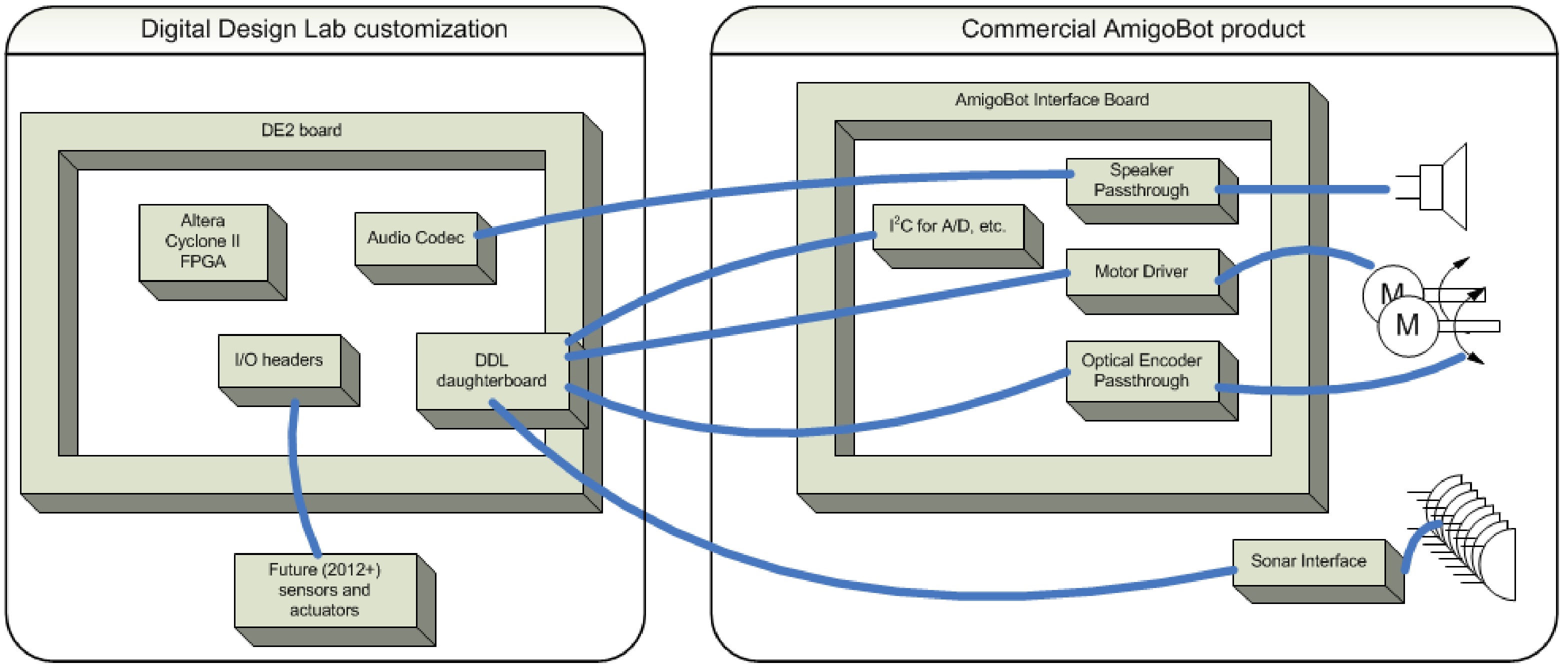
- You have now built an entire computer within the DE2 board
- Now, you will
  - Learn about using the DE2 on a robot,
  - Create an application for it, and
  - Demonstrate it

The current ECE2031 Robot

- In Summer 2010, older lab robots were gutted, adding a new internal controller board and a connected DE2 on top
- Beginning Fall 2010, capabilities have been added each semester



"AlteraBot" hardware architecture

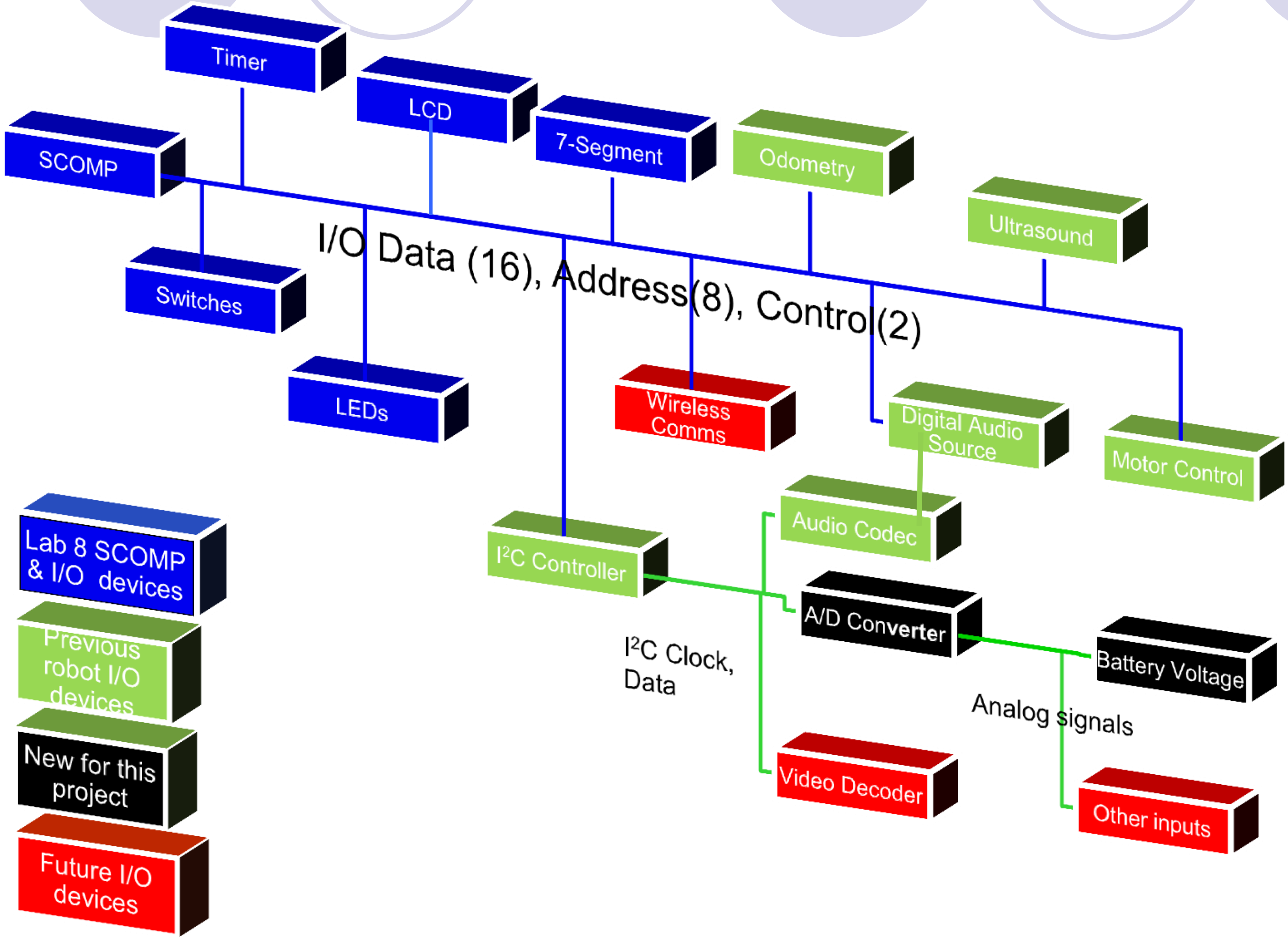


- The DE2 FPGA has direct access to robot sensors and actuators
- ECE 2031 projects add new capabilities

Past ECE2031 Robot Projects

- Velocity/position feedback from wheels
- Open loop velocity control
- Processing of sonar obstacle sensors
- Wall-following demonstration
- I^2C interface (needed for sound, battery monitoring)
- Odometry (dead-reckoning with wheel rotation sensors)
- Audio codec interface for sound output

Evolution of robot I/O subsystem



Topic for Spring 2013: Self-test

- Understand functionality of robot
- Take advantage of provided I/O devices
- Write an SCOMP program that performs a self-test of the robot, sitting in a fixed location on a desktop
- Design space –
  - Utilize the robot and its attached DE2 hardware (and possibly instructing the user to attach oscilloscope or logic analyzer)
  - Add new I/O peripherals or improve existing ones as needed
- Requirement – Application must be a program running on SCOMP, communicating with modular peripherals, using IN and or OUT commands
  - i.e., do not create VHDL modules that are “wired” anywhere but to the I/O subsystem

A common theme in past 2031 robot projects

- When something doesn't work, the robot gets blamed
- It is true – equipment DOES fail, but most problems are user-related:
  - FPGA design (possibly in bdf, possibly in VHDL)
  - SCOMP code (assembly errors can be elusive)
  - Careless errors (code not compiled, variables not initialized, something not reset)
- A simple debugging technique is to replace one component at a time with a "known-good" device
  - See Lab Manual appendix on debugging
- But how do we know a good robot?

Your Design Task for Spring 2013

- Make the DE2 board's FPGA test as much of the integrated robot/DE2 system as possible
- Your project will also include three major UPCP assignments
  - A proposal outlining what you intend to develop
  - A user manual to help anyone use your design
  - An oral presentation of your design
- You must also maintain a design logbook using forms provided by the UPCP
- One or more of the best designs will be a resource for future students
  - Chosen designs will be placed on web site as project downloads
  - And one design could be the default file loaded in all robots

Bad self-test practice

- Suppose the first thing you test is a command telling the robot to turn a wheel at a certain velocity
- If nothing happens, was it because
  - The motors never had power applied to them?
  - The motors are broken?
  - The wheel encoder (needed for velocity sensing) is broken?
  - The VHDL device that estimates velocity isn’t processing encoder signals correctly?
  - The test program was never downloaded properly to the FPGA?
- Some of these you could probably eliminate by providing verbose instructions to the user (e.g, “Did the program load?” “Did the wheel turn at all?”)
  - But you would rather minimize manual operations in a good self-test procedure
- Other possibilities could not be eliminated, because you simply are <u>testing too many things all at once</u>

Good self-test practice

- Establish communication and start testing from the FPGA outward to the DE2 board peripherals, and finally to the robot sensors and actuators.
- For example, you may want to test items in the following sequence
  - Display something that shows correct downloading of the chip (and indicates that at least part of the display is working)
  - Establish a basic communication between human user and FPGA chip. Example: User manual tells user to press a certain button, and if DE2 board displays the “right” thing, then at least that button and that display seem to be working.
  - Test functions within the FPGA, if applicable
  - Test functions involving other DE2 I/O, if applicable
  - Test battery
  - Test robot functions involving sensors (sonar, wheel encoders)
  - Test robot functions involving actuators (motors)

Example of good self-test practice

- Consider a PC and the boot “BIOS” screen
- Usually, the PC beeps first
  - That is its simplest communication to the user
  - If it doesn’t beep, you may suspect something serious is wrong
- THEN, it starts testing processor and memory
  - Sometimes, repetitive beep codes are used to communicate faults detected in the processor/memory/keyboard core system
- THEN, it detects plug and play devices and may perform basic tests on some of them
  - By this point, a video screen is assumed for user interaction, especially if display adapter is passing tests
  - Keyboard may be used to alter operation
- FINALLY, the operating system boots and performs the most advanced tests as drivers load

More good self-test practice

- Minimize the need to refer to written instructions in the user manual
  - Optimize use of LEDs, LCD, and 7-segment displays.
- When a clear failure is found, consider whether it is practical to continue testing
  - If you do not sense manual movement of wheels, it would not make sense to test the motors, for example.
  - On the other hand, if you find one bad sonar, that doesn't mean you shouldn't test them all.
- Consider the use of a "troubleshooting tree" in your user manual
  - Depending on the result of a test, you may consider alternate subsequent tests, or simply end the process with some conclusion
  - Look up "decision tree" for examples in various contexts

Optional DE2 board functions

- The DE2 board includes VGA output, keyboard input, and mouse input
- You CAN use these features, but they are advanced functions
  - We do not have time to properly discuss them in lecture
  - And you may have to add steps just to test the features themselves before using them
  - They are an inconvenience for a future user to connect
  - You might get a better grade by ignoring them and doing better tests!
- Before you choose to use them, read the relevant sections of Hamblen & Furman, and decide if you can interface them to your SCOMP (if applicable). You may even want to complete the interface BEFORE submitting your proposal.

Project details

- If it is effective, your self-test can suggest targeted use of oscilloscope and logic analyzer
  - Once a failure is detected, your user manual or user display can suggest the use of this external equipment
- You can supply gadgets or measuring devices, but it’s preferable NOT to need any <u>special</u> accessories to run your tests
  - But you probably SHOULD use <u>common</u> objects (like books or notepads to test sonar)

What should NOT be tested?

- Odometry – it requires moving the robot off the table
- Anything else that would require moving the robot from its fixed stand on the table
- External memory, IR, USB, Ethernet, video input – hard, and simply not needed

What constitutes test “failure”?

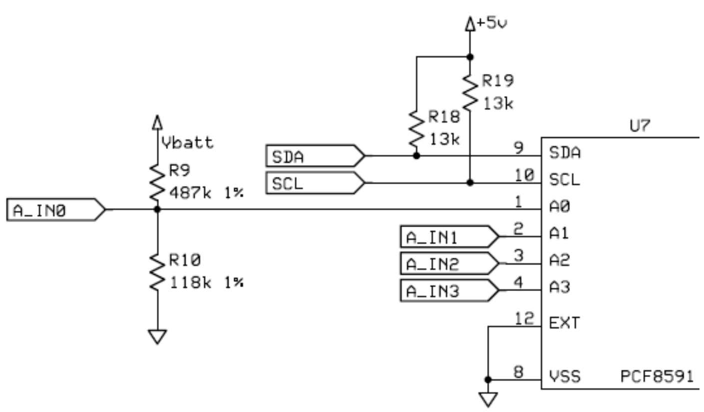
- Some specifications will be provided, such as
  - Sonar range and accuracy
  - Encoder wheel “counts” per revolution
- You can establish your own specifications where none exist, based on experience with many robots
- If you make specifications, make them such that most/all robots pass them

Should I test the battery?

- Yes. A low battery is a common cause of problems.
- Some protections are built in
  - The hardware will not allow the motors to enable when the battery level drops below about 10.8 volts
  - Yellow LED on robot will turn on when this happens
  - Battery cuts off power to the LED at an even lower level
- But the user would want to at least know the current battery voltage
- A thorough battery test requires a slow charge (many hours), followed by a discharge at normal usage rates (possibly several hours)
  - So you do not have time for such a thorough test

How do I test the battery?

- There is an analog-to-digital converter (ADC) in the robot that SCOMP can read
- Analog inputs larger than 5 V would damage ADC
- A scaled version of the battery is used instead
  - Analog input 0 is battery voltage multiplied by 118/(118+487)
  - So, for example, 12 V shows up as 2.34 V
- The ADC is one of several devices connected to the internal I2C bus



What is I2C?

- A serial bus (1 signal line, 1 clock) defined by Philips to allow integrated circuits to communicate
- Standardized hardware and communication protocol
- Recognized standard throughout the electronics industry
- Example: your smart phone has a processor that probably communicates with multiple internal devices with I2C or a similar bus like SPI

How to access I2C

- One year ago, ECE2031 students created an I2C peripheral for SCOMP
- A similar implementation is now provided to you, ready to use
- Details about how to access it, and how to use it to communicate with ADC, will be provided

Audio feedback

- You CAN generate sound, like the startup beeps in PCs
- Most of what you need will be given to you, but some of the integration with SCOMP will be left as an exercise
- More information will be provided on the project web page

Inducing failure

- Robots will generally be fully functional
- Some “bad” robots will probably be desired
- Watch for updates. We will probably have one or more robots for QUICK usage by all students, with several possible faults:
  - Variable power supply, to simulate low battery
  - One or more disconnected sonar transducers
  - Disconnected motor
  - Disconnected encoder

Project starting point

- Start with SCOMP that is provided to you
  - it will implement all instructions
  - it will have an additional DE2 I/O device working (LCD)
  - it will implement an 8-level subroutine call stack
- Modify VHDL, BDF files as needed
- Write SCASM
- If you choose to add SCOMP instructions, note that you need to change a LOCAL copy of SCASM.cfg

Project “Decision Space”

- What features to test
- How to test them
- Order in which to test them
- Modification of existing displays (LCD, 7-segment, LEDs)
- Use of displays, switches to interact with user
- Degree of “selfness” vs. requirement of user operations
- Use external test equipment?

Project phases and key dates

- Introductory exercises (March 12-14, in your regular lab section)
  - Investigate project starting point provided for you
- Brainstorm your approach and turn in proposal on April 9-11, in your lab section
- Complete your design
- Final demonstration – April 23-25
  - Make a PowerPoint presentation, explaining what worked & what didn’t.
  - Demonstrate your solution. Points for your demo will factor into your grade.
  - Turn in user manual the following Monday, April 29! (To Kevin Johnson by noon!)

Project Schedule

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		You are here →		March 7 Project background in lecture	8 Project background in lecture	9 LAB CLOSED
10 LAB CLOSED	11 OPEN HOURS	12 Pre-project Exercises & Brainstorming	13 Pre-project Exercises & Brainstorming	14 Pre-project Exercises & Brainstorming	15 Design Proposal lecture & Exam Review*	16 LAB CLOSED
17 LAB CLOSED	18 SPRING BREAK	19 SPRING BREAK	20 SPRING BREAK	21 SPRING BREAK	22 SPRING BREAK	23 LAB CLOSED
24 LAB CLOSED	25 OPEN HOURS	26 Project work	27 Project work	28 Project work	29 EXAM IN LECTURE*	30 LAB CLOSED
31 LAB CLOSED	April 1 OPEN HOURS	2 Practical Exam	3 Practical Exam	4 Practical Exam	5 Project Q&A*	6 LAB CLOSED
7 LAB CLOSED	8 OPEN HOURS	9 Project work	10 Project work	11 Project work	12 Presentation and communication tips*	13 LAB CLOSED
14 LAB CLOSED	15 OPEN HOURS	16 Project work	17 Project work	18 Project work	19 Design Report Tips*	20 LAB CLOSED
21 LAB CLOSED	22 OPEN HOURS	23 Project Demos and presentations	24 Project Demos and presentations	25 Project Demos and presentations	26 No lecture* LAB CLOSED	REPORTS DUE MONDAY NOON!

* Lecture activity on Thursday is the same

Project Demo

- Your demo will be separate from your oral (PowerPoint) presentation
  - Both done in last day of lab
- Compete head-to-head with other teams in the entire class (all lab sections)
- All section results compiled to rank teams for 500-point demo score
- Details later

Brainstorming / proposal

- Review these slides
- Get with your project team (groups of four or five)
- Use collaborative process described in the “Design Logbook” and other information provided on the UPCP web site (watch for email!)
- Come up with a technical approach and management plan
- Write proposal in the format described on the UPCP web site and in the workbook
- Your proposal will be graded like any other report for style, formatting, content, etc.

Your proposal should be detailed!

- Your proposal should explicitly describe how you address each of the items in the earlier “Decision Space” slide
  - Include some figures, such as statecharts, block diagrams
- Should describe how it will be programmed in SCOMP
  - Do not show lots of code – that comes later
  - Again, include relevant figures, such as a flowchart
- Should describe problems likely to be encountered, with ways to address them
  - Include backup plans if a high-risk task fails
- Assign task responsibilities as you decide what is most interesting to each team member
- More on this next week in lecture

Experiment before proposing!

- During your first project day in the lab, conduct these activities
  - Investigate the design file template provided
  - Drill down into the details of the devices
    - I/O decoder
    - SCOMP (with 8-level stack, all commands)
  - Watch for posted exercises on project download page
- It is never too early to prototype some ideas for your approach
  - What might be too hard?

Prelab activities for next week

- Last Prelab Quiz will cover
  - Chapter 15 of textbook
  - These slides
- Follow instructions in email from Kevin Johnson (will be sent Friday afternoon) regarding logbook and brainstorming
  - You will not need the lab to complete this but will need to print a few pages.

Clarifications

- Additional clarifications will be posted on the web site, or as direct answers to email
  - When a general question is asked, everyone gets copied on the anonymized response