CSCE 4114
Embedded Systems

Top Down Structured Design

David Andrews
Rm 527 JBHT
dandrews@uark.edu

What We Will Cover Today

- Typical Design Flow
  - Top Down Design Approach
- Understanding Requirements
  - Functional
  - Behavioral
  - Timing
  - Physical
- Perform Requirements Analysis on Specific Example
  - Taking Verbal Description and Generating Requirements

Overview of Top Down Design

- Requirements Assessment: *<u>What</u>* is it
  - Paper and pencil work/Negotiations with customer
  - Final requirements is your contract with customer, need to get it correct
  - Documentation: System Requirements Specification (SRS)
- Top Level Design: *<u>Where</u>* are you going to put it
  - Functional Descriptions/Block Diagrams
  - Hardware Software partitioning
  - Automated Tools available (will discuss later)
  - Documentation: Hardware/Software Top Level Design Documents (STLDD/HTLDD)

Overview of Top Down Design

- Detailed Design: *How* are you going to design it
  - Hardware Component selections, Fan Ins/Outs, simulations
  - Software Modules, Subroutines
  - Documentation: Hardware/Software Detailed Design Document (HDDD/SDDD)
- Integration and Test: *Verification* Frustration
  - Before Modules Brought Together, Unit Test
  - Bring Modules Together. This is where the rubber hits the road
  - Testing Based on Pre-Defined Hardware/Software Test Plan (HTP/STP)

Top Down Philosophy

- Design Process is iterative, you make a stab at next lower level, then based on results, revisit upper level and adjust. Adjustments affect everyone, not just you

- This seems like work, why not just "go for it"
  - Need to know what you are designing first before designing it.
  - Much easier to get a warm and fuzzy that the big picture is correct
  - Most projects are group oriented, you need to interface with others

  - 10:1 rule Each hour spent at the higher abstract level will save 10 hours at the next lower level.
  - Very expensive in Cost and Time to get to integration and test, and find that you make errors.

Requirements Analysis

- Requirements Assessment: What is it
  - Timing: How fast does your system need to be ? MIPS/FLOPS, turnaround times, input/output times etc
  - Sizing: Most Systems are size limited. Anyone can develop a supercomputer/flight controller etc if you have enough space.
  - Interfaces: Are you hooked up to sensors inputting data ? How is the world going to communicate with your system ?
  - Other Special Requirements (Radiation Hardened, etc) These can affect cost, size, performance etc.
- Customer may give you a laundry list that sometimes can be conflicting. You need to apply engineering expertise to give honest assessments. Customer may not really know all requirements, you again must help
- Most Contract bids are based on Requirements Analysis. You must have a good understanding of all requirements in order to propose a feasible system solution

- THIS IS YOUR CONTRACT, WHEN YOU IMPLEMENT ALL REQUIREMENTS YOU ARE DONE. IF THEY ARE NOT IMPLEMENTED, YOU ARE NOT

Top Level Design

- Top Level Design: Where are you going to put it
  - System Block Diagram
  - System Address Map
  - Debug Support
  - Derived Requirements
  - Subsystem Interfaces
- This is your Top Level Partitioning. Teams are assigned to implement modules defined here.
- IMPORTANT: Interfaces are defined. This allows teams to work independently and simultaneously.
- Derived requirements are targets for teams. They may not know or care about overall system. They just meet the derived requirements.
- Don't miss debug support. This may not be discussed in requirements analysis, but is key for further design and implementation

Detailed Design

- Each Module in Top Level Design is further partitioned and designed. More derived requirements for each block in a module.
- Interfaces within module are defined, chips selected, actual signals/interconnections are defined.
- Functional Simulations are performed to guarantee the functionality of the Module. I.e., is the correct answer produced ? Are the algorithms correct ?
- Simulations of Hardware performed in structured fashion (More Later) with automated tools.
- Detailed Design is last chance to functionally knock out bugs before tedious implementation.

Automated Tools are Helping in the Detailed Design

Implementation

- At this stage, the design should be proven correct. You want to implement the correct logic, etc.
- Circuit design based on Logic Family
- Board Layouts.
- Physical constraints such as size, weight, power must be met.
- This is the last step in the design process (before trying to make sense of what you designed). It took a long time to get here, but if done correctly, this is only a mechanical excersize.

Integration and Test

- Where the Rubber hits the road.....

- This step is usually underwhelmed in planning stage, and is overwhelming in actual work.
- Integration and Test can take as long or longer than the other design steps.
- Philosophy: Minimize the unknowns. Common approach of junior engineers is to "go for it". Better to take tiny bites than choke on a big piece.
- Test Modules First, subsystems second, then two subsystems, then multiple tested subsystems etc. You must go back and retest other components when anything that affects it changes.
- The results of this is when you get paid........

Summary

- Design Methods
  - Top Down
  - Bottom Up
  - Random
- Top Down Design Steps
  - System Requirements Specification **What**
    - Document SRS
  - Top Level Design **Where**
    - Document STLDD/HTLDD
  - Detailed Design **How**
    - Document (Schematics, Code)
  - Integration and Test **Selloff**
    - Document STP/HTP

Requirements

Lets Take a Closer Look at Requirements