What is an Embedded System?

- A type of computer system
- Traditional Definitions
  - Limited in hardware and software vs the PC
  - Designed to perform a dedicated function
  - Designed to perform functions on a wide variety of general purpose functions
- What’s Changes
  - Embedded microprocessor have become more powerful
  - Embedded systems now perform multi-functioned tasks

Embedded System Usage

- Automotive
  - Antilock brakes, Ignition systems, Engine Control, Cruise, etc.
- Consumer Electronics
  - TV, appliances, toys, cell phones, robotic sweepers, etc.
- Industrial Control
  - Robotics, PLC, Human Machine Interfaces (HMI), Robotics, Drives
- Networking
  - Wireless routers, hubs, search engines, gateways
- Medical
  - Infusion pumps, dialysis machines, prosthetic devices, cardiac monitors
- Office Automation
  - Fax machines, copiers, printers, scanners,

Design Challenges

1. Defining and capturing the design of a system
2. Cost Limitations (low profit margin must sell millions)
3. Determining system integrity; such as reliability and safety
4. Working within the confines of the available functionality (memory, battery life, processing power, etc.)
5. Marketing and selling the device
Development Models

1. **Big-Bang Model**: little to no planning or processes in place before or during the development of the system

2. **Code-and-Fix Model**: product requirements are defined but no formal processes are in place before the start of the development

3. **Waterfall Model**: there is a process of developing a system in steps, where the results of one step flow into the next step

4. **Spiral Model**: there is a process of developing a system in steps, and throughout the various steps feedback is obtained and incorporated back into the process

Embedded system Design and Development Lifecycle Model (Phase 1: Detail)
DE2-115: Programming Block Diagram

DE2-115: Display output

DE2-115: What is Nois II?

- Altera’s Second Generation Soft-Core 32 Bit RISC Microprocessor
- Can Be Targeted For All Altera FPGAs
- Synthesis Using Quartus II Integrated Synthesis

DE2-115: Nois II Block Diagram
VHDL Review

- VHDL: an industry standard for digital circuits
  - Very High Speed Integrated Circuits
  - Hardware
  - Description
  - Language
- Differences from traditional programming languages
  - Inherently parallel
  - Mimics the behavior of digital systems
  - Allows for the incorporation of timing requirements
  - Describes a system as an interconnects of components

VHDL Basic Syntax

- Basic operators
  - <= operator for setting values
  - ; terminator statement
  - -- indicates a comment
- Keywords (few examples)
  - and, or, not, after
  - when, if, else, then -- conditionals
- Variable declaration
  - Not case-sensitive
  - May use numbers, letters, underscore (no spaces)
  - Must start with a letter and cannot end with an underscore

VHDL Basic Structure

- Consists of 2 major parts
  - entity declaration: defines the I/O of the models
    - entity declaration
      - entity entity-name
      - port (interface-signal-declaration)
      - end entity entity-name
    - where interface-signal-declaration is a list of signals, mode and signal_type
      - i.e. port(signal-name(s)): mode signal_type
    - mode: describes the direction data is transferred:
      - in to the port, out of the port, inout bi-directional, or buffer out with feedback
    - signal_type: describes the data type
      - bit (std_logic) single signal having values of 0 or 1
      - bit-vector (std_logic_vector) bus signals that have values 0 or 1*
      - *vectors must have their range specified i.e. (3 downto 0)
  - architecture body: describes the operation of the model
    - architecture declaration
      - architecture architecture-name of entity-name is
      - Declarations here local I/O definitions (in gates and other components)
    - component component-name
      - port (interface-signal-declaration)
      - end component component-name
    - begin
      - architecture-body
      - end architecture architecture-name
Library and Packages: A library is a place where the compiler stores information about a design project. A VHDL package is a file or module that contains declarations of commonly used objects, data type, component declarations, signal, procedures and functions that can be shared among different VHDL models.

For example, std_logic is defined in the package ieee.std_logic_1164 in the ieee library. Thus, in order to use the std_logic one needs to specify the library and package at the beginning of the VHDL file using the `library` and the `use` keywords:

```
library ieee;
use ieee.std_logic_1164.all;
```

The `.all` extension indicates to use all of the ieee.std_logic_1164 package.

VHDL Code: 4-bit Full Adder

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY adder IS
PORT ( Cin : IN STD LOGIC;
      X, Y : IN STD LOGIC VECTOR(3 DOWNTO 0);
      S : OUT STD LOGIC VECTOR(3 DOWNTO 0);
      Cout : OUT STD LOGIC);
END adder;

ARCHITECTURE Structure OF adder IS
  SIGNAL C : STD LOGIC VECTOR(1 TO 3);
  COMPONENT fulladd
    PORT ( Cin, X, Y : IN STD LOGIC;
           S, Cout : OUT STD LOGIC);
  END COMPONENT;
BEGIN
  stage0: fulladd PORT MAP ( Cin , X(0), Y(0), S(0), C(1) ) ;
  stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
  stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
  stage3: fulladd PORT MAP ( X(3), Y(3), Cin , S(3), Cout ) ;
END Structure;
```

Full Adder Example

```
-- Example of a four bit adder
library ieee;
use ieee.std_logic_1164.all;

entity FULLADDER is
  port (a, b, c: in std_logic;
        sum, carry: out std_logic);
end FULLADDER;
architecture fulladder_behav of FULLADDER is
begin
  sum <= (a xor b) xor c;
  carry <= (a and b) or (c and (a xor b));
end fulladder_behav;
```

4-bit Adder Example

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
entity FOURBITADD is
  port (a, b: in std_logic_vector(3 downto 0);
       Cin : in std_logic;
       sum: out std_logic_vector (3 downto 0);
       Cout, V: out std_logic);
end FOURBITADD;
architecture fouradder_structure of FOURBITADD is
  component FULLADDER
    port (a, b, c: in std_logic;
          sum, carry: out std_logic);
  end component;
begin
  FA0: FULLADDER port map (a(0), b(0), Cin, sum(0), c(1));
  FA1: FULLADDER port map (a(1), b(1), C(1), sum(1), c(2));
  FA2: FULLADDER port map (a(2), b(2), C(2), sum(2), c(3));
  FA3: FULLADDER port map (a(3), b(3), C(3), sum(3), c(4));
  V <= c(3) xor c(4);
  Cout <= c(4);
end fouradder_structure;
```