

Hardware Diversity Training Kit

- NRC Funded Project
- D3 Training for non-nuclear engineering students
- Diversified hardware to beat CCF/CMF
- Hardware kit is developed
- Ready for class activity

“nuclear renaissance” and “Nuclear Crisis”

- More electricity from NPP
 - Power Upgrades
 - New Construction and Operating License (or “Combined License” [COL]) Applications
 - 18 COL applications for 28 new reactor units (as of 6/30/2010)
 - 104 Nuclear Reactors in 31 States
 - No New Reactor Since 1978
- Digital/Computer Control of NPP
 - Critical Importance of Reliability of H/W and S/W
 - Challenging issues of I&C, Security, Licensing, Safety Interfaces
- Safety?
 - More important now with Japanese NPPs

Defense-in-Depth

- **NPPs generally heavily reliant on Redundancy**
 - Components still fail anyway
 - Failure could be dangerous
 - Resiliency by Redundancy
 - Diversity over Redundancy due to CMF
- **Instrumentation and Control**
 - Analog → Digital
 - Failures in Computers due to CCF
 - How do you protect against Failure?
 - Is having multiple Enough?
 - Importance of Independence/Diversity

NRC sponsored Project of Nuclear Science course development

- Objective
 - Hardware (Equipment) Diversity against CMF in H/W and S/W
 - Exposure to Nuclear Science and Engineering
 - Exposure and familiarity to Hardware and Software and Safety issues of Digital I&C in NPP
- Target
 - All Non-nuclear engineering students at Howard University
 - First Offering: Fall 2011 (Course title: Computers and Nuclear Energy)
- **U.S. NRC Education Grant: No. NRC-27-10-1123**

Diversity

- What is Diversity?
 - *Diversity* is the use of two or more mutually exclusive means of performing the same function.
 - *Diversity* is a principle in instrumentation systems of sensing different parameters, using different technologies, using different logic or algorithms, or using different actuation means to provide several ways of detecting and responding to a significant event.
 - *Diversity types*: Human diversity, design diversity, **software diversity**, functional diversity, signal diversity, and **equipment diversity**.
- How to make diversified Computer Hardware Kit?
 - Build a Kit of Four 'diverse' architectures

Diversity in Computers

- Diversity in the kit
 - FPGA, Harvard, PIC, and, PLA
- Is the kit really diverse or just redundant?
 - Criteria for diversity (in Computer Systems)
 - Different manufacturers of fundamentally different designs
 - Same manufacturer of fundamentally different designs
 - Different manufacturers making the same design
 - Different versions of the same design

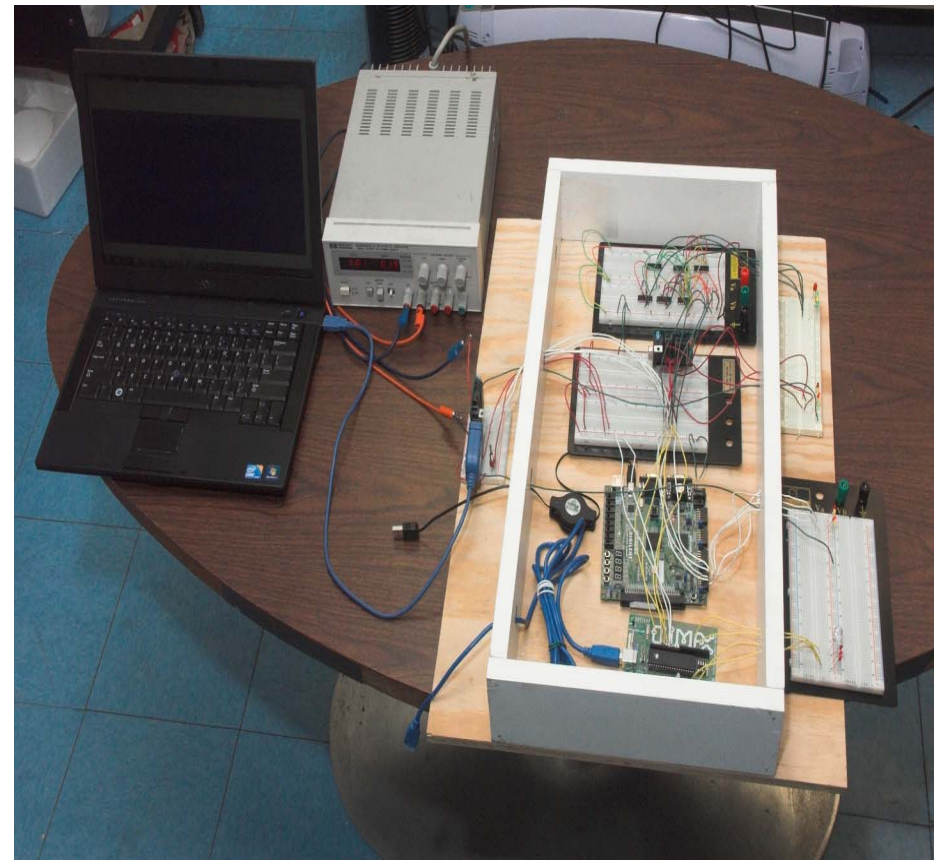
Training Kit

- Equipment diversity is the use of different equipment to perform similar safety functions.
- “Different” means sufficiently unlike as to significantly decrease vulnerability to common failure.

Architecture	Manufacturer	Compiler/ Interpreter	Language
PIC- Peripheral Interface Controller	Microchip	MPLAB	Assembly Language
BS 2 – BASIC Stamp 2	Parallax	BASIC	Basic Editor
FPGA - Field- programmable Gate Array	Digilent	XILINX	VHDL

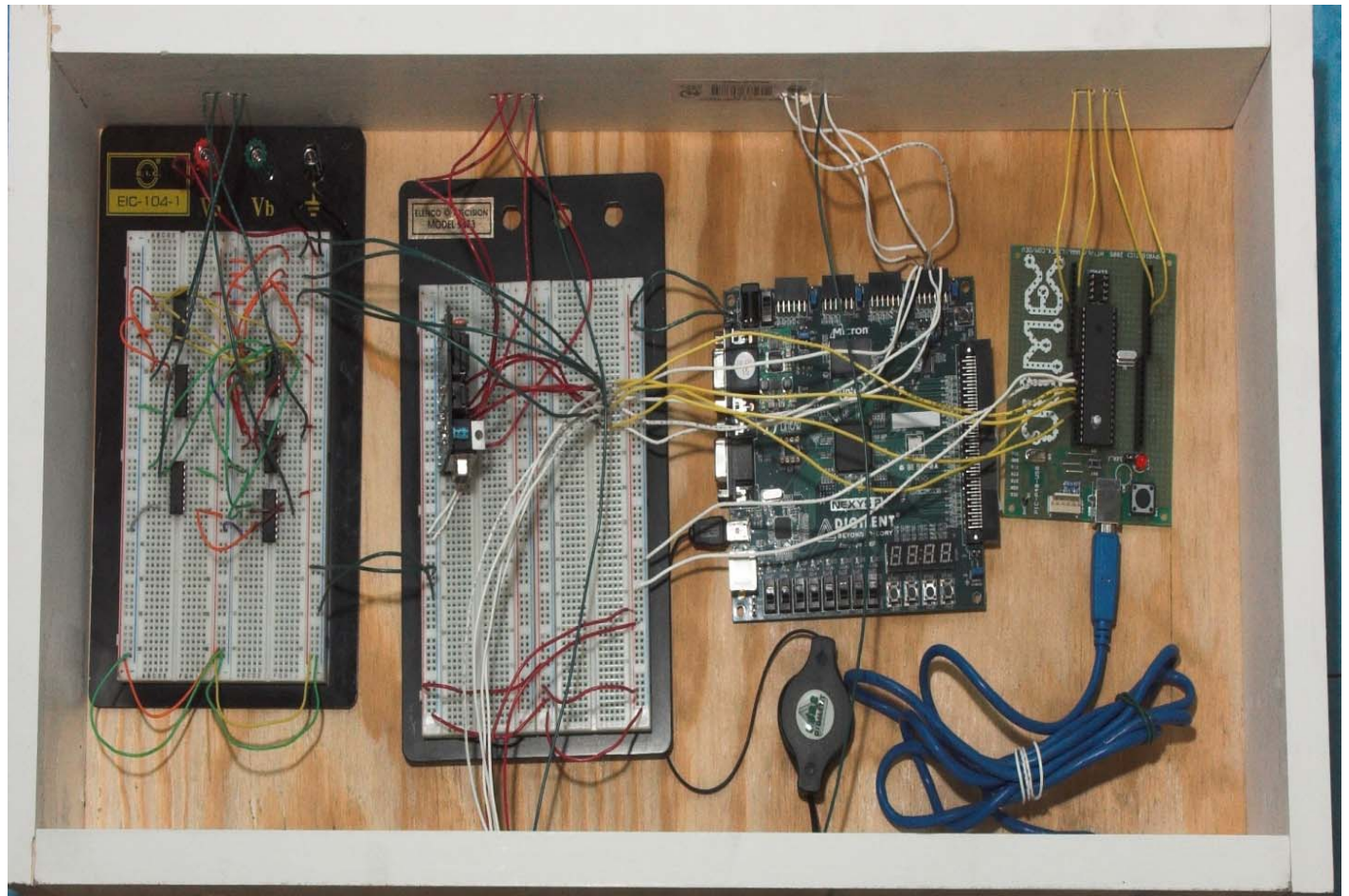
Training Kit

- Completely different architectures: one fails, the other works (Independent)
- **Introducing CCF into Set-up:**
 - **Power** (Power spikes/ surges)
 - **Humidity/ Moisture** (Unusual moisture build up)
 - **External Magnetic Field**



Implementation

- PLA
 - BS 2
 - FPGA
 - PIC
-
- Another BS 2 for
Instrumentation
Simulation/
Scenario
Generation



Scenario Based Testing

INPUTS

- Pressure
- Temperature
- Water Level (First three for the reactor)
- Power (at the pumps)

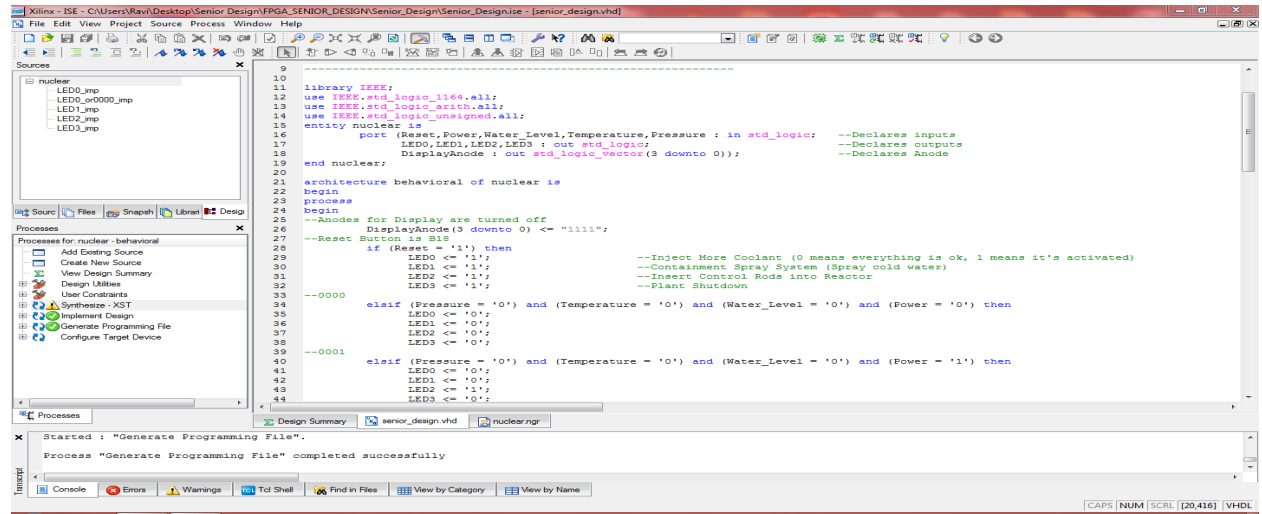
OUTPUTS

- LED 1 - Injection of more coolant
- LED 2 - Containment spray system (Spray Cold water containing Boron)
- LED 3 - Scram control (Drop Zircon Rods)
- LED 4 is (Injection of Neutron Poison Solutions into the plant)



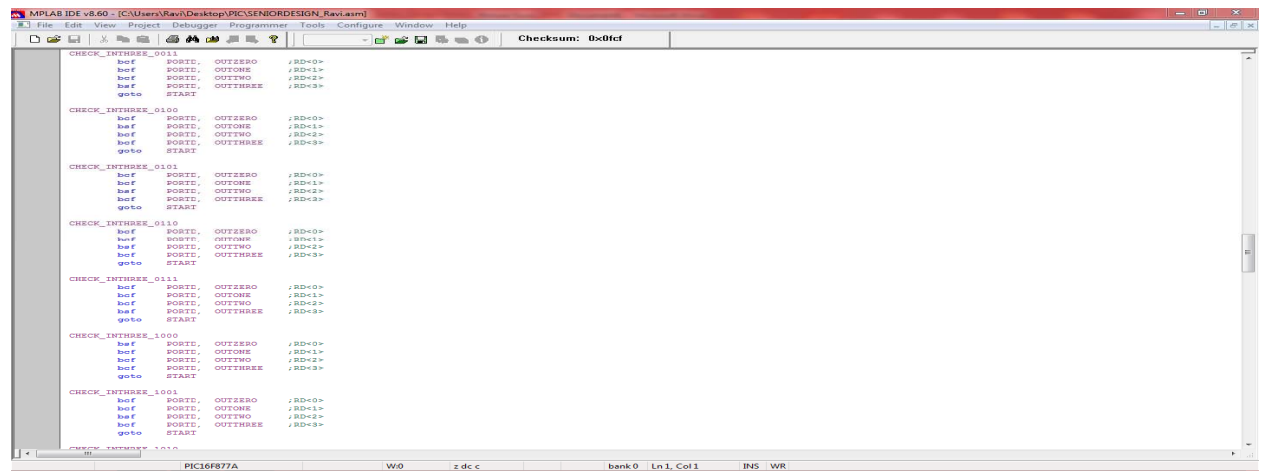
Different Coding Environments

- NEXYS 2
DIGILIENT FPGA
(XILINX)
- PIC (MPLAB)



The screenshot shows the Xilinx ISE IDE interface. The main window displays a VHDL code file named 'senior_design.vhd'. The code defines a 'nuclear' entity with inputs 'Reset', 'Power', 'Water_Level', 'Temperature', and 'Pressure', and outputs 'LED0', 'LED1', 'LED2', 'LED3', and 'DisplayAnode'. The architecture is behavioral and includes logic for displaying anode states and controlling LEDs based on sensor inputs. The left pane shows the 'Sources' window with a project tree for 'nuclear' containing files like 'LED0_0000.vhd', 'LED1_0000.vhd', 'LED2_0000.vhd', 'LED3_0000.vhd', and 'senior_design.vhd'. The bottom pane shows the 'Processes' window with a list of tasks including 'Add Existing Source', 'Create New Source', 'View Design Summary', 'Design Utilities', 'Use Constraints', 'Synthesize -XST', 'Implement Design', 'Generate Programming File', and 'Configure Target Device'. The 'Generate Programming File' process is highlighted, and a status bar at the bottom indicates 'Started: "Generate Programming File". Process "Generate Programming File" completed successfully'.

```
10
11 library IEEE;
12 use IEEE.std_logic_1164.all;
13 use IEEE.std_logic_arith.all;
14 use IEEE.std_logic_unsigned.all;
15 entity nuclear is
16     port (Reset, Power, Water_Level, Temperature, Pressure : in std_logic; --Declares inputs
17           LED0, LED1, LED2, LED3 : out std_logic; --Declares outputs
18           DisplayAnode : out std_logic_vector(3 downto 0)); --Declares Anode
19 end nuclear;
20
21 architecture behavioral of nuclear is
22 begin
23     process
24     begin
25         --Anodes for Display are turned off
26         DisplayAnode(3 downto 0) <= "1111";
27         --Reset Button is B18
28         if (Reset = '1') then
29             LED0 <= '1'; --Inject More Coolant (0 means everything is ok, 1 means it's activated)
30             LED1 <= '1'; --Containment Spray System (Spray cold water)
31             LED2 <= '1'; --Insert Control Rods into Reactor
32             LED3 <= '1'; --Plant Shutdown
33         --0000
34         elsif (Pressure = '0') and (Temperature = '0') and (Water_Level = '0') and (Power = '0') then
35             LED0 <= '0';
36             LED1 <= '0';
37             LED2 <= '0';
38             LED3 <= '0';
39         --0001
40         elsif (Pressure = '0') and (Temperature = '0') and (Water_Level = '0') and (Power = '1') then
41             LED0 <= '0';
42             LED1 <= '0';
43             LED2 <= '1';
44             LED3 <= '0';
45         end if;
46     end process;
47 end;
```



The screenshot shows the MPLAB IDE v8.60 interface. The main window displays assembly code for a PIC16F877A. The code consists of multiple sections, each starting with a label like 'CHECK_INTERR_0111', 'CHECK_INTERR_0100', 'CHECK_INTERR_0101', 'CHECK_INTERR_0110', 'CHECK_INTERR_0111', 'CHECK_INTERR_1000', and 'CHECK_INTERR_1001'. Each section contains instructions for setting port directions (PORTD, PORTC, PORTB, PORTA) and setting output values (OUTZERO, OUTONE, OUTTWO, OUTTHREE). The code is organized into a loop structure with 'goto START' instructions. The bottom status bar shows the target device as 'PIC16F877A' and the current location as 'W0 zdc c bank0 Ln1.Col1 INS WR'.

```
CHECK_INTERR_0111
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START

CHECK_INTERR_0100
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START

CHECK_INTERR_0101
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START

CHECK_INTERR_0110
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START

CHECK_INTERR_0111
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START

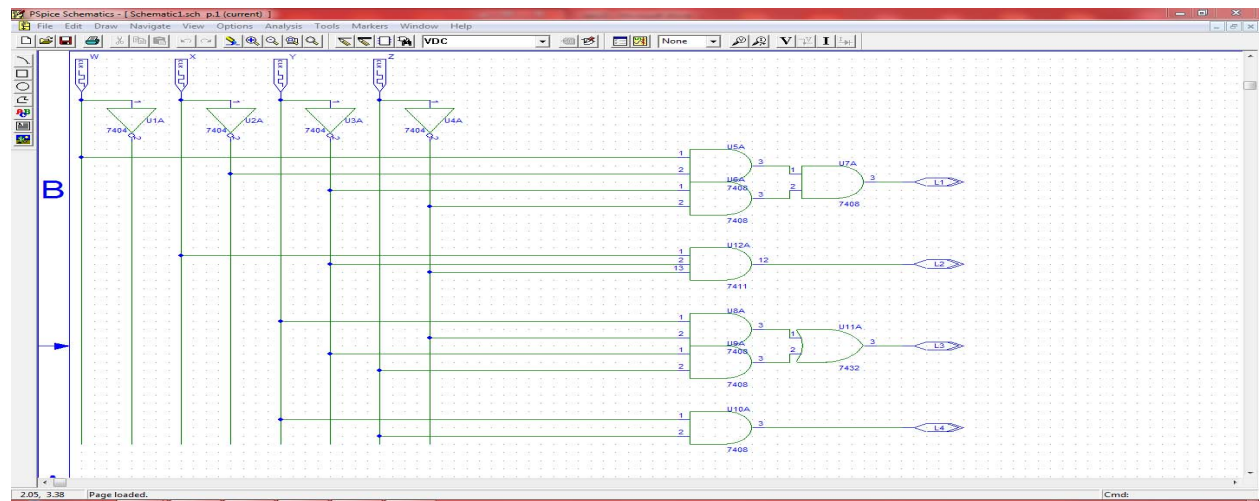
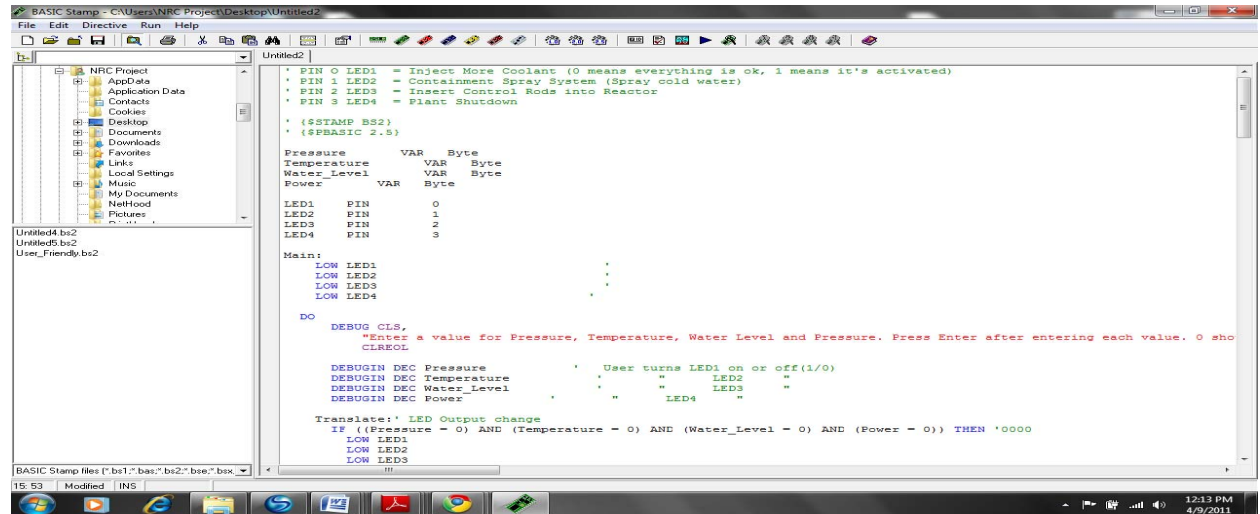
CHECK_INTERR_1000
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START

CHECK_INTERR_1001
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START

CHECK_INTERR_1010
    bsf    PORTD, OUTZERO    ;RD=0>
    bsf    PORTC, OUTONE     ;RD=1>
    bsf    PORTB, OUTTWO     ;RD=2>
    bsf    PORTA, OUTTHREE   ;RD=3>
    goto   START
```


Different Coding Environment

- PARALLAX BASIC Stamp[®] 2 (Basic Stamp 2 Editor)
- PLA (PSPICE)



Class Activity

- Form Different Teams based on background and strength
 - FPGA, BS2, Logic, and PIC programming Teams
- Coding practice
 - Tutorial for those who do not have background
- Testing with a CCF
- Learn the lessons from the test