Digital Design and Verification Training

Getting Started

Agenda

- Introductions
- Course Overview
- Administrative Information
- Basics of C

Trainers: EE Department UET, KICS

Dr. Muhammad Tahir
Professor, Department of Electrical Engineering, UET Lahore

Mr. Umer Shahid
Lecturer, Department of Electrical Engineering, UET Lahore

Mr. Rehan Naeem
Lecturer, Department of Electrical Engineering, UET Lahore

Trainee's Introduction

- Now it's your turn

Training Overview

- Training objectives
  - Develop a strong foundation in digital design principles and computer architecture
  - Bridge the gap between academic knowledge and industry requirements
  - Cultivate practical skills in digital design tools and methodologies
  - Foster teamwork and communication skills essential for professional engineering environments

What you can Expect

- 1 month of coaching ~ 10k a month
  - Week-1: Productivity Tools
  - Week-2: Digital Systems Design
  - Week-3: Computer Architecture
  - Week-4: Digital Systems Verification
- Hands on Projects ~ 10k a month
  - Complex datapaths and controllers
  - RISCV Microprocessor System Design
  - Verification
  - Software
  - Just like you’re working in a company

What is Expected of You

- Attendance
- Performance in Multiple Evaluations
    - Knowledge
    - Skill
    - Speed

What is Expected of You

- Top-notch Communications
  - Daily Updates
    - What you have done today.
    - What you’re going to do tomorrow.
    - Are there any roadblocks?
  - Clarity and Accuracy
    - Spoken and Written
- Independence
  - Task assignment to task completion
  - Doesn’t stop you from collaboration

Timings

- Monday to Friday, 09.00 am to 04.00 PM
- Coaching
  - 09.00 am to 12.00 pm
  - Will include a couple of small breaks.
  - May end earlier than the allocated time.
- Hands on Lab Sessions
  - 12.00 pm to 04.00 pm

What's Next

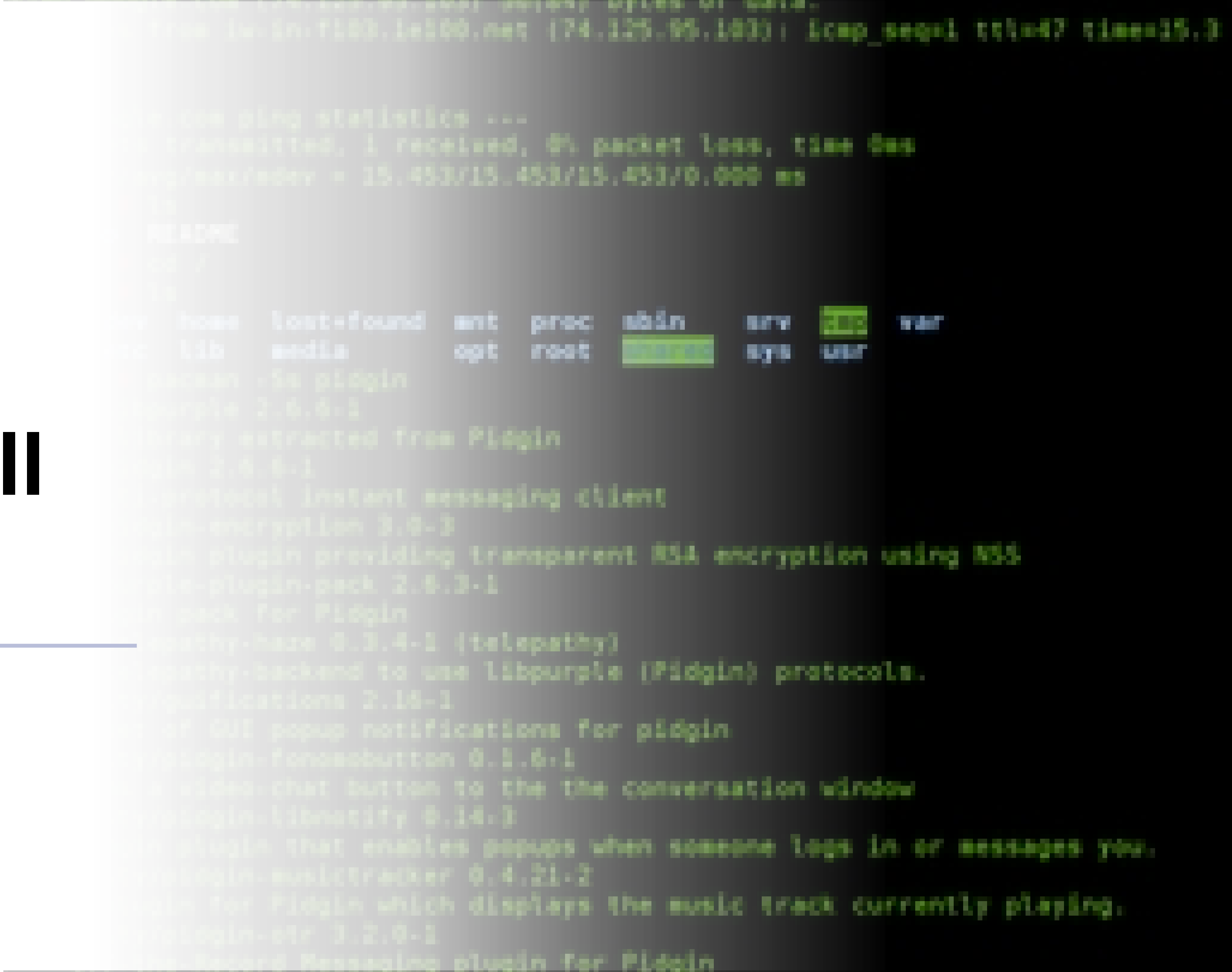
- Internship - 30k to 40k a month
    - Xcelerium’s Proprietary Projects
- Full-Time Job - 130k to 150k a month

Administrative Information

- Reporting procedure
- Slack information
- https://join.slack.com/t/xceleriumdigi-t6v9440/shared_invite/zt-3bif7sic2-YyHNbEca0DAqMuGnqK86iQ

Linux Shell Scripting

Digital Design and
Verification Training



Linux Shell Scripting

- A shell is a program that commands the operating system to perform actions.
- You can enter commands in a console on your computer and run the commands directly, or you can use scripts to run batches of commands.
- Shells like PowerShell and Bash give system administrators the power and precision they need for fine-tuned control of the computers they're responsible for.

Shell Commands

- The full syntax for a Bash command is:

```
command [options] [arguments]
```

- Which options and arguments varies from command to command. To learn about the options for a command, use the man (for "manual") command.

```
man <command>
```

File and Directory Commands

- Make directory
  - `mkdir <name>`
- Navigating – `cd <option>`
  - `cd ~`
  - `cd ..`
- List Contents
  - `ls <name>`
- Removing directory
  - if empty – `rmdir <name>`
  - Not empty – `rm –r <name>`
- Copy files or directories
  - `cp <source> <dest>`
- Move or rename files or directories
  - `mv <source> <dest>`

Text Processing and File Commands

- Creating
  - `touch <file-name>`
- Concatenate and display file content
  - `cat <file-name>`
- Search text using patterns
  - `grep <pattern>`
- Replace Text in a file:
  - `sed <text.to.replace> <file_name>`

System Information

- Print system information: `uname`
- Display Linux processes: `top`
- Report a snapshot of the current processes: `ps`

Copy command – cp

- Copy files and directories

```
cp <name-to-be-copied> <name-of-the-copy>
```

  - if you use the -i (for "interactive") flag, Bash warns you before deleting existing files.

```
cp -i hello.txt bye.txt
```

  - copy all the files in the current directory to a subdirectory subdir1

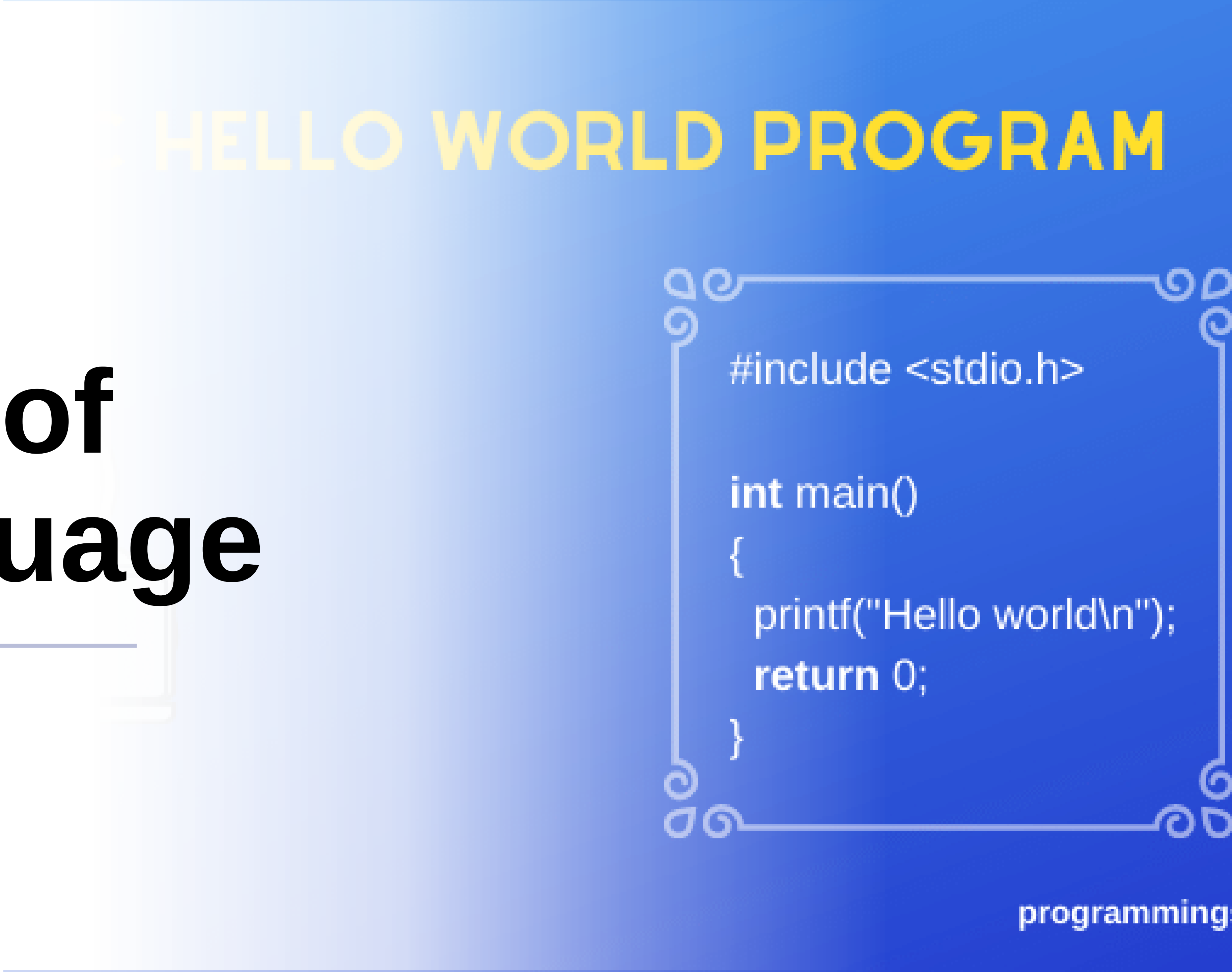
```
cp * subdir1
```

  - To copy all the files in a subdirectory named subdir1 into a subdirectory named subdir2

```
cp subdir1/* subdir2
```

Basics of C Language

Digital Design and
Verification Training



```
#include <stdio.h>

int main()
{
  printf("Hello world\n");
  return 0;
}
```

Quick Revision - C Language Essentials

- Basic syntax and structure
- Data types and variables
- Operators and expressions
- Control structures (if, switch, loops)
- Functions
- Arrays and strings

Basic Syntax and Structure

- Structure of a C program:

```
#include <stdio.h> // Preprocessor directive
int main()
{ // Main function
// Your code here
return 0; }
```

Basic Syntax and Structure

- Basic data types

```
int age = 25;
float pi = 3.14;
double precise_pi = 3.14159265359;
char grade = 'A';
```

- Size of data types

```
printf("Size of int: %zu bytes\n", sizeof(int));
printf("Size of float: %zu bytes\n", sizeof(float));
printf("Size of double: %zu bytes\n", sizeof(double));
printf("Size of char: %zu bytes\n", sizeof(char));
```

Basic Syntax and Structure

- Operators and Expressions

```
int a = 10, b = 3;
printf("Sum: %d\n", a + b); // 13
printf("Difference: %d\n", a - b); // 7
printf("Product: %d\n", a * b); // 30
printf("Quotient: %d\n", a / b); // 3
printf("Remainder: %d\n", a % b); // 1
```

- Increment/Decrement

```
int x = 5;
printf("x: %d\n", x++); // Prints 5, then x becomes 6
printf("x: %d\n", ++x); // x becomes 7, then prints 7
```

Basic Syntax and Structure

- If-else example

```
int score = 95;
if (score >= 90)  { printf("Grade: A\n"); }
else if (score >= 80) { printf("Grade: B\n"); }
else if (score >= 70) { printf("Grade: C\n"); }
else { printf("Grade: F\n"); }
```

Basic Syntax and Structure

- Switch example

```
char grade = 'B';
switch (grade) {
case 'A':
    printf("Excellent!\n");
   break;
case 'B':
   printf("Good job!\n");
   break;
```

```
case 'C':
   printf("Average
performance.\n");
   break;
default:
    printf("Need
improvement.\n"); }
```

Basic Syntax and Structure

- For loop

```
for (int i = 0; i < 5; i++)
{ printf("%d ", i); } // Output: 0 1 2 3 4
```

- While loop

```
int n = 1;
while (n <= 5) {
printf("%d ", n * n);
n++; } // Output: 1 4 9 16 25
```

Basic Syntax and Structure

- Function definition and call:

```
int square(int x)
{ return x * x; }
int main() {
int num = 4;
printf("Square of %d is %d\n", num, square(num));
return 0;
}
```

Basic Syntax and Structure

- Array initialization and access

```
int numbers[] = {1, 2, 3, 4, 5};
for (int i = 0; i < 5; i++)
{ printf("%d ", numbers[i]); }
```

Basic Syntax and Structure

- String operations

```
char str1[20] = "Hello";
char str2[] = " World";
strcat(str1, str2);
printf("Concatenated string: %s\n", str1);
printf("Length: %zu\n", strlen(str1));
```

Basic Demo

`#include <stdio.h>`
`#include <string.h>`
`#define MAX_NAME_LENGTH 50`
`void greet(char* name)`
`{ printf("Hello, %s!\n", name); }`
`int main() {`
`char name[MAX_NAME_LENGTH];`
`printf("Enter your name: ");`
	`fgets(name, MAX_NAME_LENGTH, stdin);`
	`name[strcspn(name, "\n")] = 0; // Remove newline`
	`greet(name);`
	`printf("Let's count to 5:\n");`
	`for (int i = 1; i <= 5; i++)`
	`{ printf("%d ", i); }`
	`printf("\n");`
	`return 0; }`