

ACTIVITES D'ENSEIGNEMENT

Responsabilité de 5 UE en Master:

MASTER M2: 2 U.E: -Circuiterie CMOS et Analyse temporelle (TIME -12H cours)
 -Architectures numériques (ANUM -16 H cours)
 -Conception en vue du test (DFT- 12H)

MASTER M1: 2 UE: -Traitement du signal (8H)
 -Traitement numérique du signal (60H Cours TD/TME)

LICENCE L1 : Initiation à la programmation impérative en C (42H TD/TME)

LICENCE L2: Structures des données avancées en C (42H TD/TME) (2006-2007)

Encadrement: projets et stages MASTER (M1 et M2)

Responsable des stages pro du Master (en moyenne une trentaine d'étudiants annuellement)

DETAIL DES UE :

Niveau L1 : UE Initiation à la programmation impérative en C

Introduction

Cet enseignement présente le langage le plus répandu actuellement, C, en abordant les bases de la programmation impérative et de ce langage, tout en familiarisant les étudiants avec le système Linux et les outils standards de développement dans cet environnement.

Cette UE a un poids de 6 ECTS et s'étend sur 12 semaines.

Description

Cette UE constitue une introduction à la programmation impérative en C.

L'objectif est de débiter la programmation impérative en C tout en permettant aux étudiants de se familiariser avec l'environnement de travail choisi. Sont traités : les types, les expressions, les structures de contrôle, les fonctions et leurs modes de passages de paramètres.

Les TME seront réalisés dans un environnement Linux : introduction au système de fichiers, utilisation d'outils standards d'édition et de compilation.

Bibliographie

- Le langage C: Norme ANSI. Brian W. Kernighan & Dennis M. Ritchie, Dunod, 2004
- Programmer en langage C. Claude Delannoy, Eyrolles, 2002
- C: A Reference Manual. Samuel P. Harbison & Guy L. Steele Jr., Prentice Hall, 2002
- C: A Software Engineering Approach. Peter A. Darnell & Philip E. Margolis, Springer, 1996

•

Contenu indicatif par semaine :

- **Semaine 1:** cours : environnement Unix + C
fichier source, exécutable, qu'est-ce que compiler, comment sont rangées les données, printf
TME : environnement Unix, quelques commandes et compilation d'un premier programme C
- **Semaine 2:** expressions, types, types de base, types prédéfinis, printf, if...then...else...
- **Semaine 3:** boucles
- **Semaine 4:** fonctions
- **Semaine 5:** passage de paramètres
- **Semaine 6:** tableaux statiques plus petit élément, produit scalaire, recherche dans un tableau non trié, dans un tableau trié, produit matriciel
- **Semaine 7:** tableaux suite, debugger

- **Semaine 8:** récursivité
- **Semaine 9:** tris (sélection, insertion, fusion)
expérimentation avec tableau aléatoire, tableau rangé en ordre croissant et décroissant
- **Semaine 10:** fin des tris
- **Semaine 11:** enregistrements
- **Semaine 12:** fichiers et formats

Niveau L2: UE Programmation impérative et structures de données en C

Introduction

Bien que créé en 1972, le langage C fait encore aujourd'hui l'objet d'une très grande popularité et de très nombreuses applications sont décrites en C. Il est : universel, compact, structuré, proche de la machine, efficace, mais il peut aussi être dangereux car il n'impose que très peu de contraintes au programmeur. En suivant cette UE vous apprendrez à maîtriser ce formidable outil.

Cette UE a un poids de 6 ECTS. Elle s'étend sur 12 semaines.

Description

L'objectif principal de cette UE (accompagnée des UE "Initiation à la programmation impérative en C" et "Programmation impérative et concurrente") est la maîtrise complète de la programmation impérative. Associée au module "Initiation à la programmation impérative en C" cette UE permettra aux étudiants de maîtriser un langage informatique de "référence".

Cette UE présentera sous un angle impératif les structures de données les plus couramment utilisées en programmation (listes, piles et arbres). Le point sera particulièrement mis sur la gestion explicite de la mémoire (pointeurs et allocation dynamique). Les principes de la compilation séparée seront présentés en cours et pratiqués en TME (.h, .c, .o et Makefile). Les accès aux fichiers seront abordés ainsi que le lien avec d'autres langages et quelques notions d'algorithmique et de complexité.

Les TME seront réalisés dans un environnement Linux standard en utilisant les outils les plus couramment répandus : gcc, ddd.

La connaissance des structures de base du C (types, structures de contrôle et fonctions) est très fortement recommandée.

Bibliographie

Brian.W. Kernighan, Dennis M. Ritchie. Le langage C, Norme ANSI. Dunod, 2004.

Claude Delannoy. Programmer en langage C. Eyrolles, 2002, Collection

Contenu indicatif par semaine

Semaine 1: Syntaxe du C n°1 : les types et les expressions, outils du C n°1 (gcc, ddd).

Semaine 2: Syntaxe du C n°2 : les fonctions, passage de paramètres, outils du C n°2 (.h, .c, make).

Semaine 3: Les pointeurs, les structures, l'allocation dynamique.

Semaine 4: Exemples de programmation utilisant pointeurs et structures.

Semaine 5: Les listes chaînées.

Semaine 6: Les arbres d'expression, initiation au graphisme.

Semaine 7: Les tables de hachage.

Semaine 8: Présentation du projet Scrabble.

Semaine 9: Scrabble 2.

Semaine 10: Scrabble 3.

Semaine 11: Scrabble fin.

Semaine 12: Conclusion et ouverture vers la programmation concurrente.

MASTER 1 SESI : UE Traitement numérique du signal

Introduction

Ce module vise d'une part à présenter une introduction des concepts fondamentaux de la théorie du signal et des principaux algorithmes et opérateurs utilisés pour le traitement numérique. Il vise,

ensuite, à présenter l'architecture interne des processeurs spécialisés pour le traitement du signal (DSP), ainsi que les techniques de programmation de ces processeurs avec des exemples pratiques. Cette UE a un poids de 6 ECTS et s'étend sur 10 semaines.

Description

La plupart des systèmes intégrés utilisés dans les applications télécom ou dans les applications vidéo contiennent des processeurs spécialisés pour le traitement numérique du signal (Digital Signal Processing), pour des fonctions de filtrage et/ou de mise en forme. Ce module vise d'une part à présenter les bases de la théorie du traitement du signal et les principaux algorithmes et opérateurs utilisés pour le traitement numérique du signal. Il vise par ailleurs à présenter l'architecture interne des processeurs spécialisés pour le traitement du signal (DSP), ainsi que les techniques de programmation de ces processeurs. Il comprend 3 parties : Eléments de traitement de signal
Implantation matérielle : Algorithmes et Opérateurs de base
Les processeurs DSP : Architecture et Programmation

Bibliographie

LES PROCESSEURS DE TRAITEMENT DU SIGNAL

BAUDOIN G., VIROLLEAU F. Paris : Dunod, 1996.

Digital Signal Processing Principles, Algorithms, and Applications John G. Proakis et Dimitris G. Manolakis Troisième édition, 1996 Prentice-Hall ISBN : 0-13-373762-4

DSP Processor Fundamentals Architectures and Features Phil Lapsley, Jeff Bier, Amit Shoham et Edward A. Lee 1997, IEEE Press

DSP Applications Using C and the TMS320C6x DSK (Hardcover) by Rulph Chassaing, edit. J. WILEY

Contenu indicatif par semaine

Semaine 1: Introduction/Rappel au traitement du signal: éléments de base.

Semaine 2: Systèmes de numération. Représentations des nombres en virgule fixe et virgule flottante. Algorithmes d'addition et architectures dédiées. Critères de performances et Comparaisons. Convertisseurs entre systèmes de numération.

Semaine 3: Algorithmes de multiplication, Implantations matérielles. Critères de performances et Comparaisons. Etude de l'unité de base : le multiplieur accumulateur. Proposition d'une architecture performante.

Semaine 4: Processeur FFT. Algorithmes en bases deux et quatre. Algorithmes de décomposition pour les grands nombres d'échantillons. Unités arithmétiques de base. Architectures de processeurs FFT. Différentes alternatives. Applications mono-dimensionnelles et bidimensionnelles pour le traitement d'image.

Semaine 5: Les processeurs DSP. Particularités : jeux d'instructions, modes d'adressage, ALUs, etc. Principaux domaines d'applications. Les évolutions des architectures et de la complexité des DSP. Architectures parallèles. Problèmes de génération de code optimisé.

Semaine 6: Choix d'un processeur. Présentation détaillée de l'architecture et du jeu d'instructions.

Semaine 7: Implantation d'un système à base de DSP. Environnement logiciel. Génération de code. Test et debug. Mise en œuvre d'une application.

Semaine 8: Techniques d'optimisation de code. Application au calcul d'un produit scalaire et au filtrage numérique.

Semaine 9: Mise en œuvre de la FFT sur DSP. Optimisation de l'implantation en fonction des algorithmes. Etude des critères tels que l'occupation de l'espace mémoire, calcul d'adresses, complexité du contrôle etc. Comparaison des performances avec l'implantation matérielle.

Semaine 10: Mise en œuvre de la convolution sur DSP. Différents Algorithmes pour les applications mono-dimensionnelles et bidimensionnelles. Etude d'exemples en traitement d'image.

Moyens demandés et innovations pédagogiques

Les moyens à utiliser sont les outils de simulation type MATHLAB et un Kit d'évaluation à base de systèmes DSP comprenant un environnement de développement et de debug et une carte DSP raccordée à un ordinateur hôte qui est ici le PC.

MASTER 2 SESI: UE Architectures pour le traitement numérique (Hardware architecture for arithmetic)

Introduction

Cette UE fondamentale présente différentes techniques architecturales pour le calcul arithmétique intensif et des exemples d'applications réelles utilisant ces techniques.

Cette UE a un poids de 3 ECTS et s'étend sur 7 semaines.

Description

Dans la première partie du cours, on présente les algorithmes de traitement arithmétique des données et les techniques architecturales utilisées pour leurs implantations. La seconde partie est consacrée à l'étude des choix d'implantation pour quelques structures qui apparaissent souvent lors de l'implantation d'algorithmes du traitement de l'information (signal, images, données). Le but pour chacune des 3 dernières séances est d'étudier une fonction « importante » du domaine des télécommunications du point de vue de son implantation matérielle : son contexte, la raison d'une implantation matérielle et les aspects critiques associés à une telle implantation, l'analyse des choix d'architecture qui se présentent pour cette fonction, etc.

Bibliographie

- J.M. Muller, Arithmétique des Ordinateurs, Masson, Paris, 1989
- Milo D. Ercegovac, Tomás Lang Digital Arithmetic (The Morgan Kaufmann Series in Computer Architecture and Design) (Hardcover)
- Simulateur en ligne d'Alain GUYOT

Contenu indicatif par semaine

- **Semaine 1:** Systèmes de numération. Passage d'un système à un autre. Algorithmes et architectures pour l'addition binaire. Comparatif performances et complexités
- **Semaine 2:** Algorithmes et architectures pour la multiplication. Comparatif performances et complexités
- **Semaine 3:** Algorithmes et architectures pour la division et la racine carrée. Comparatif performances et complexités
- **Semaine 4:** Systèmes de numération pour le calcul en virgule flottante. Norme IEEE 754. Algorithmes et architectures pour l'addition et multiplication. Unité flottante IEEE. Conversions entre entiers et flottants.
- **Semaine 5:** Etude de structures de calcul couramment utilisées dans le domaine des télécommunications.
- **Semaine 6:** Compression vidéo (algorithme, étude architecturale, implantation)
- **Semaine 7:** Codage canal (algorithme, étude architecturale, implantation)

MASTER 2 SESI: UE Analyse temporelle des circuits numériques (Digital circuits timing analysis)

Introduction

Acquisition d'une compréhension approfondie du fonctionnement des circuits CMOS, ainsi que de la méthode générale permettant d'analyser leurs performances temporelles.

Cette UE a un poids de 3 ECTS et s'étend sur 7 semaines.

Description

Ce cours traite des éléments de base permettant de concevoir les circuits en technologie CMOS. Il se repose sur une modélisation simplifiée des dispositifs tant en fonctionnement statique qu'en fonctionnement dynamique. Une analyse détaillée des contraintes temporelles est effectuée. Elle est suivie par des propositions des solutions et techniques utilisées pour l'accroissement des performances.

Bibliographie

- Principles of CMOS VLSI Design, A Systems Perspective, 2nd edition, Neil H. E. Weste et Kamran Eshraghian, Addison-Wesley, 1993
- Annaratone, M., Digital CMOS Circuit Design, Kluwer Academic Publishers, Boston, 1986.
- Banzhaf, W., Computer-Aided Circuit Analysis using SPICE, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

- Fabricius, E. D., Introduction to VLSI Design, McGraw-Hill International Editions, New York, 1990.

Contenu indicatif par semaine

- **Semaine 1:** Rappels sur les dispositifs MOS. Topologie et modélisation statique et dynamique de ces dispositifs. Régimes de fonctionnement. Technologie CMOS. Paramètres technologiques. Modélisation statique détaillée de l'inverseur CMOS. Distribution des alimentations.
- **Semaine 2:** Modélisation dynamique de l'inverseur CMOS. Etude des temps de montée, de descente et de propagation. Règles de base. Amplification des signaux, techniques utilisées, distribution des horloges. Puissance dissipée.
- **Semaine 3:** Familles logiques CMOS. Etude comparative. Analyse temporelle. Equilibre des arbres.
- **Semaine 4:** Eléments séquentiels en CMOS. Règles et contraintes temporelles (temps d'établissement, de maintien, de propagation, de cycle etc.). Analyse temporelle d'un réseau complet (combinatoire et séquentiel).
- **Semaine 5: TP1 :** analyse statique
- **Semaine 6: TP2 :** analyse dynamique
- **Semaine 7: TP3 :** analyse temporelle (extraction chemins critiques), analyse de métastabilité

Moyens demandés et innovations pédagogiques

Simulateurs électriques SPICE et ELDO.

MASTER 2 SESI: UE Conception en vue du test (Design for testability)

Introduction

Cette UE vise à présenter les techniques de conception permettant de faciliter le test de production des circuits intégrés numériques.

Cette UE a un poids de 3 ECTS et s'étend sur 7 semaines.

Description

Elle traite essentiellement des modèles de fautes et en particulier le modèle de collage, la simulation de fautes et analyse de taux de couverture. Les Techniques de Scan-path, du boundary Scan et BIST (Build In Self Test) sont également présentées.

Contenu indicatif par semaine

- **Semaine 1:** Test et testabilité, généralités
- **Semaine 2:** Génération des vecteurs de test (ATPG)
- **Semaine 3:** Techniques de conception en vue du test (DFT)
- **Semaine 4:** Test intégré (BIST)
- **Semaine 5:** TP Familiarisation avec l'outil Tetramax
- **Semaine 6:** TP Génération de vecteurs de test et simulation de fautes
- **Semaine 7:** TP Test intégré (BIST)