Board of Studies

Course Proposal Template

PROPOSED COURSE TITLE: Computer Architecture and Design

Suggested acronym: CARD

PROPOSER(S): Nigel Topham, Vijay Nagarajan, Boris Grot

DATE: 25th September 2018

Version 1.0

SUMMARY

This template contains the following sections, which should be prepared roughly in the order in which they appear (to avoid spending too much time on preparation of proposals that are unlikely to be approved):

[Guidance in square brackets below each item. Please also refer to the guidance for new course proposals at http://www.inf.ed.ac.uk/student-services/committees/board-of-studies/course-proposal-guidelines. Examples of previous course proposal submissions are available on the past meetings page http://web.inf.ed.ac.uk/infweb/admin/committees/bos/meetings-directory.]

SECTION 1 – CASE FOR SUPPORT

[This section should summarise why the new course is needed, how it fits with the existing course portfolio, the curricula of our Degree Programmes, and delivery of teaching for the different years it would affect.]

1a. Overall contribution to teaching portfolio

[Explain what motivates the course proposal, e.g. an emergent or maturing research area, a previous course having become outdated or inappropriate in other ways, novel research activity or newly acquired expertise in the School, offerings of our competitors.]

This is an invited proposal to merge two existing UG3 courses, Computer Design (INFR09046, acronym CD) and Computer Architecture (INFR09009, acronym CAR), and forms part of the overhaul of Informatics courses primarily to reduce the number of courses offered.

The CD and CAR courses contain some overlap of material, and therefore make a logical pairing as candidates for merging into one course. In addition, some material currently taught in CD and CAR is also taught (albeit in less detail) in INF2C-CS. This naturally means the resulting merged course can and should omit some material from each of the two original courses, and can assume a higher level of background knowledge from INF2C-CS than is currently the case in CD and CAR. It is anticipated that some of the more advanced material from CAR will migrate to a UG4 course, but some CD material will have to be dropped completely from our syllabus.

The resulting 20-point level 10 course, entitled **Computer Architecture and Design**, will deliver an essential and core component of the undergraduate computer systems syllabus. The merger will reduce the number of taught courses by 1, and the total points taught will be reduced from 30 to 20.

1b. Target audience and expected demand

[Describe the type of student the course would appeal to in terms of background, level of ability, and interests, and the expected class size for the course based on anticipated demand. A good justification would include some evidence, e.g. by referring to projects in an area, class sizes in similar courses, employer demand for the skills taught in the course, etc.]

This course will appeal to a broad cross-section of Informatics students. The syllabus of this new course is designed not only for students wishing to specialize in computer systems, but as a mid-level course for students of software engineering and AI who need to know more about the underlying hardware of their computing platforms.

In common with the preceding CD course, this course is designed for a target audience without prior experience in computer design and, most importantly, without the need for any prior knowledge in electronics.

INF2C-CS, or equivalent, will be a prerequisite for this course. This will allow the new course to avoid repeating the introductory computer architecture material presented in INF2C-CS.

There are a significant number of students each year who follow up their UG3 CD or CAR courses with a closely-related project that relies on these UG3 courses. It is therefore important that we deliver this course primarily to the UG3 class.

Our UG3 students often remark that they would choose to take either Computer Design or Computer Architecture, but that two courses in a somewhat similar area is too much. We

therefore expect the class size for this new course to be larger than CD or CAR. We also expect that the broader scope of this course will also attract students who might not have taken either CD or CAR in previous years. Estimating numbers is difficult, but our best estimate is a class size of between 40 and 80 students.

1c. Relation to existing curriculum

[This section should describe how the proposed course relates to existing courses, programmes, years of study, and specialisms. Every new course should make an important contribution to the delivery of our Degree Programmes, which are described at http://www.drps.ed.ac.uk/17-18/dpt/drps_inf.htm.

Please name the Programmes the course will contribute to, and justify its contribution in relation to courses already available within those programmes. For courses available to MSc students, describe which specialism(s) the course should be listed under (see http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2017/programme-guide/specialist-areas), and what its significance for the specialism would be. Comment on the fit of the proposed course with the structure of academic years for which it should be offered. This is described in the Year Guides linked from http://web.inf.ed.ac.uk/infweb/student-services/ito/students.]

Undergraduate Programmes

- Artificial Intelligence and Computer Science (BSc Hons) (UTAICSC)
- Computer Science (BEng Hons) (UTCMPSIBE)
- Computer Science (BSc Hons) (UTCMPSI)
- Computer Science and Electronics (BEng Hons) (UTCMPSE)
- Computer Science and Physics (BSc Hons) (UTCMPPH)
- Informatics (MInf) (UTINFMT)
- Software Engineering (BEng Hons) (UTSWENG)

Postgraduate Programmes

- Computer Science (MSc) (Full-time) (PTMSCCMPSI1F)
- Computer Science (MSc) (Part-time) - 2 Years (PTMSCCOMSC1P)
- Computer Science (MSc) (Part-time) - 3 Years (PTMSCINFEM2P)
- High Performance Computing (MSc) (Full-time) (PTMSCHPCMP1F)
- High Performance Computing (MSc) 3 Years (Part-time) (PTMSCHPCMP3P)
- Informatics (MSc) (Full-time) (PTMSCINFMT1F)
- Informatics (MSc) (Part-time) - 2 Years (PTMSCINFMT3P)
- Informatics (MSc) (Part-time) - 3 Years (PTMSCINFMT2P)

The justification for this course is that it replaces two existing courses, each of which is already making an important contribution to these programmes.

The natural place for this course is in the UG3 year, as it sits naturally between INF2C and a number of more advanced UG4 courses. There have been suggestions that it should be taught only in alternate years. However, that would prevent this course feeding in to advanced UG4 courses and final year projects, which CD and CAR do at present. This proposal is for a course that is normally run each year (subject to the usual resourcing constraints for annual courses).

1d. Resources

[While course approvals do not anticipate the School's decision that a course will actually be taught in any given year, it is important to describe what resources would be required if it were run. Please describe how much lecturing, tutoring, exam preparation and marking effort will be required in steady state, and any additional resources that will be required to set the course up for the first time. Please make sure that you provide estimates relative to class size if there are natural limits to its scalability (e.g. due to equipment or space requirements). Describe the profile of the course team, including lecturer, tutors, markers, and their required background. Where possible, identify a set of specific lecturers who have confirmed that they would either like to teach this course apart from the proposer, or who could teach the course in principle. It is useful to include ideas and suggestions for potential teaching duty re-allocation (e.g. through course sharing, discontinuation of an existing course, voluntary teaching over and above normal teaching duties) to be taken into account when resourcing decisions are made.]

TBD

SECTION 2 – COURSE DESCRIPTOR

[This is the official course descriptor that will be published by the University and serves as the authoritative source of information about the course for student via DRPS and PATH. Current course descriptions in the EUCLID Course Catalogue are available at www.euclid.ed.ac.uk under 'DPTs and Courses', searching for courses beginning 'INFR'*]*

2a. Course Title *[Name of the course.]*:

Computer Architecture and Design

2b. SCQF Credit Points:

[The Scottish Credit and Qualifications Framework specifies where each training component provided by educational institutions fits into the national education system. Credit points per course are normally 10 or 20, and a student normally enrols for 60 credits per semester. For those familiar with the ECTS system, one ECTS credit is equivalent to 2 SCQF credits. See also http://www.scqf.org.uk/The%20Framework/Credit%20Points.]

20

SCQF Credit Level:

[These levels correspond to different levels of skills and outcomes, see http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf At University level, Year 1/2 courses are normally level 8, Year 3 can be level 9 or 10, Year 4 10 or 11, and Year 5/MSc have to be level 11. MSc programmes may permit a small number (up to 30 credits overall) of level 9 or 10 courses.]

10

Normal Year Taken: 1/2/3/4/5/MSc

[While a course may be available for more than one year, this should specify when it is normally taken by a student. "5" here indicates the fifth year of undergraduate Masters programmes such as MInf.]

3

Also available in years: 1/2/3/4/5/MSc

Different options are possible depending on the choice of SCQF Credit Level above: for level 9, you should specify if the course is for 3rd year undergraduates only, or also open to MSc students (default); for level 10, you should specify if the course is available to 3rd year and 4th year undergraduates (default), 4th year undergraduates only, and whether it should be open to MSc students; for level 11, a course can be available to 4th and 5th year undergraduates and MSc students (default), to 5th year undergraduates and MSc students, or to MSc students only]

4, 5, and MSc

Undergraduate or Postgraduate?

[If the course is only available to MSc students, then it must be classified as a Postgraduate course. All other courses, regardless of level, will be classified as Undergraduate]

Undergraduate

2c. Subject Area and Specialism Classification:

[Any combination of Computer Science, Artificial Intelligence, Software Engineering and/or Cognitive Science as appropriate. For courses available to MSc students, please also specify the relevant MSc specialist area (to be found in the online MSc Year Guide at http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2017/programme-guide/specialist-areas), distinguishing between whether the course should be considered as "core" or "optional" for the respective specialist area.]

Computer Science (core) Computer Science and Electronics (optional) Computer Science and Physics (optional) MSc (Computer Systems, Software Engineering & High-Performance Computing - optional)

Appropriate/Important for the Following Degree Programmes:

[Please check against programmes from http://www.drps.ed.ac.uk/17-18/dpt/drps_inf.htm to determine any specific programmes for which the course would be relevant (in many cases, information about the Subject Area classification above will be sufficient and specific programmes do not have to be specified). Some courses may be specifically designed for non-Informatics students or with students with a specific profile as a potential audience, please describe this here if appropriate.]

Relevant to the following undergraduate programmes

- Artificial Intelligence and Computer Science (BSc Hons) (UTAICSC)
- Computer Science (BEng Hons) (UTCMPSIBE)
- Computer Science (BSc Hons) (UTCMPSI)
- Computer Science and Electronics (BEng Hons) (UTCMPSE)
- Computer Science and Physics (BSc Hons) (UTCMPPH)
- Informatics (MInf) (UTINFMT)
- Software Engineering (BEng Hons) (UTSWENG)

Relevant to the following taught postgraduate programmes

- Computer Science (MSc) (Full-time) (PTMSCCMPSI1F)
- Computer Science (MSc) (Part-time) - 2 Years (PTMSCCOMSC1P)
- Computer Science (MSc) (Part-time) - 3 Years (PTMSCINFEM2P)
- High Performance Computing (MSc) (Full-time) (PTMSCHPCMP1F)
- High Performance Computing (MSc) 3 Years (Part-time) (PTMSCHPCMP3P)
- Informatics (MSc) (Full-time) (PTMSCINFMT1F)
- Informatics (MSc) (Part-time) - 2 Years (PTMSCINFMT3P)
- Informatics (MSc) (Part-time) - 3 Years (PTMSCINFMT2P)

Timetabling Information:

[Provide details on the semester the course should be offered in, specifying any timetabling constraints to be considered (e.g. overlap of popular combinations, other specialism courses, external courses etc).]

This course has no known timetabling constraints and could be delivered in either semester.

Timetabling should avoid placing too many UG3 computer systems courses in the same semester, in order to balance the workloads across semesters for students who commonly take these courses as part of their Computer Science degree programmes.

2d. Summary Course Description:

[Provide a brief official description of the course, around 100 words. This should be worded in a student-friendly way, it is the part of the descriptor a student is most likely to read.]

In this course you will learn how to design a computer and understand the performance characteristics of computers. You will first acquire a working knowledge of digital design, through the Verilog Hardware Description Language, along with a good theoretical grounding in the design of the key components of a microprocessor. You will have an opportunity to learn, both theoretically and practically, how the Quantitative Approach to computer architecture enables computer architects to analyse and optimize microprocessors to maximize performance. Along the way, you will spend time in the lab, designing real hardware, and later in the course you will apply your recently-acquired knowledge of quantitative computer architecture to analyse a simulated system and optimize its performance.

Course Description:

[Provide an academic description, an outline of the content covered by the course and a description of the learning experience students can expect to get. See guidance notes at: http://www.studentsystems.is.ed.ac.uk/staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

Academic Description

Building on the summary description, a more in-depth, academic description of the learning aims, nature and context of the course.

This new course presents a logical re-factoring of a sub-set of the material previously contained in the UG3 Computer Architecture and Computer Design courses. The aim of this course is to give students a comparatively deep understanding of computer architecture, to an intermediate level, together with a solid understanding of techniques used to design the logical building block from which a computer is constructed. We consider an intermediate level in computer architecture to extend up to the point where students have a good understanding of instruction set architecture, single-issue in-order pipelined execution of instructions, superscalar out-of-order execution, and the memory hierarchies required by those processors. Within a processor, we explore the principles and practice of arithmetic and logic unit design, of the caches from which memory hierarchies are constructed, and the memory and logic gate technologies from which computers are constructed. Throughout the course, there is a strong emphasis on the

Quantitative Approach to computer architecture; this informs not only the theoretical topics but also the practical assignments, which always embody some element of the quantitative approach.

The philosophy of this course is that learning about computer architecture is particularly effective if reinforced by implementing key aspects of processor design, in real hardware when feasible, but also at higher levels of abstraction using simulated systems. This approach has been used very effectively in the previous Computer Design and Computer Architecture courses, and feedback often cites the value placed on the lab exercises by students.

Outline Contents

A more detailed outline content or syllabus (where this is convention within the discipline).

N.B. it is advisable not to be overly prescriptive such as indicating in which weeks of the semester certain topics will be taught in the course descriptor since this is likely to change annually. This level of detail should be articulated in the course handbook if/as required.

- **Fundamentals**
  - **Review of logic design and implementation technologies**; from simple combinational logic to state machines for sequential circuits; logic design using Verilog and introduction to FPGAs.
  - **Register Transfer Level design principles**; registers, clocks, timing budgets, setup and hold margins, clock skew, clock-domain crossing and synchronization, metastability.
  - **Quantitative computer architecture**; performance evaluation methods and metrics, principles of high-performance design.
- **Processor Architecture**
  - **Instruction Set Architecture (ISA) design**; instruction set classes, registers, memory addressing. RISC vs CISC, how the ISA supports high-level languages, quantitative approach to ISA design. Example ISAs (e.g. MIPS, RISC-V). ISA requirements for embedded systems.
  - **Pipelined processor design**; pipeline hazards and interlocks, control prediction techniques and their usage.
  - **Out-of-order execution**; scoreboards, reservation stations, register renaming, quantitative analysis of performance.
- **Computer Arithmetic and ALU Design**
  - **Introduction to binary arithmetic functions**; fixed-point addition, subtraction, multiplication and division.
  - **Advanced techniques in computer arithmetic**; carry-lookahead adders, parallel-prefix adders, Booth-coded multipliers, Wallace and Dadda trees, sub-word parallelism, fractional fixed-point multiply-accumulate operations.
  - **Floating-point computations**; IEEE standard, floating-point addition and multiplication, high-performance fused-multiply-add architectures.
- **Memory System Design**
  - **Memory hierarchies**; review of principles, quantitative analysis of memory hierarchy performance; exploring the design space of cache parameters.
  - **Cache coherence in multi-core architectures**; protocols and implementation techniques.
  - **Main memory design**; Interfacing between processor and memory, synchronous and asynchronous bus protocols.
  - **Error detection and correction schemes**; parity, Hamming codes, SECDED.

Student Learning Experience

A narrative description of how the course will be taught, how students are expected to engage with their learning and how they will be expected to evidence and demonstrate their achievement of the intended learning outcomes.

The course will be taught through a combination of conventional lectures, scheduled laboratory practical sessions, tutorials, informal feedback sessions, and private study.

Students are expected to spend a significant amount of time in the first two weeks reading around the subject, using references supplied (particularly the main course text for the computer architecture material).

Students will be expected to attend three-hour scheduled lab sessions during weeks 3-6, during which they will complete two assessed exercises. They will be expected to design a digital system, implement it in the Verilog language, simulate it, map it to an FPGA, and demonstrate a working system to one of the demonstrators. They will also be expected to give a verbal explanation of how their design works, and walk the demonstrator through their Verilog code explaining the salient features. Demonstrators will also instruct students in best practices for digital design, during the labs. Students will submit write-ups for their lab designs, for which specific questions will be provided in the handout – those questions will elicit responses that test the expected learning outcomes.

When lab sessions are running there will also be weekly examples sheets distributed to students (but no tutorials). Students will have the option to work through the questions in their own time, submit solutions to the lecturer, and come along to a feedback sessions to discuss their work. These will be mostly small problem-solving questions based on that week's lecture material. Solutions will be made available the following week.

When the non-lab coursework is being undertaken there will be weekly tutorials, based around a set of tutorial worksheets. These will reinforce the material (mostly in the computer architecture part of the course) and provide topics of discussion for tutors and tutees. The non-lab coursework will involve some element of quantitative computer architecture design and evaluation, which will be assessed via a written report. Guidance on report structure will be given in the handout, and will be designed to elicit responses that test the students' achievements against the learning outcomes.

Pre-Requisite Courses:

[Specify any courses that a student must have taken to be permitted to take this course. Pre-requisites listed in this section can only be waived by special permission from the School's Curriculum Approval Officer, so they should be treated as "must-have". By default, you may assume that any student who will register for the course has taken those courses compulsory for the degree for which the course is listed in previous years.

Please include the FULL course name and course code].

Course Title: Informatics 2C – Computer Systems (or equivalent)

Course Code: INFR08018

Co-Requisite Courses:

[Specify any courses that should be taken in parallel with the existing course. Note that this leads to a timetabling constraint that should be mentioned elsewhere in the proposal. Please include the FULL course name and course code].

None

Prohibited Combinations:

[Specify any courses that should not be taken in combination with the proposed course. Please include the FULL course name and course code].

None

Other Requirements:

[Please list any further background students should have, including, for example, mathematical skills, programming ability, experimentation/lab experience, etc. It is important to consider that unless there are formal prerequisites for participation in a course, other Schools can register their students onto our courses, so it is important to be clear in this section. Also be aware that MSc students are unlikely to have the pre-requisite courses, so alternative knowledge should be recommended. If you want to only permit this by special permission, a statement like "Successful completion of Year X of an Informatics Single or Combined Honours Degree, or equivalent by permission of the School." can be included.]

Successful completion of an external course that is equivalent to the prerequisite INF2C course, by permission of the School. Students should be competent programmers, as coursework may require students to create software. There are no specific programming language requirements

Available to Visiting Students: Yes/No

[Provide a justification if the answer is No.]

Yes

2e. Summary of Intended Learning Outcomes (MAXIMUM OF 5):

[List the learning outcomes of the course, emphasising what the impact of the course will be on an individual who successfully completes it, rather than the activity that will lead to this outcome. Further guidance is (apparently not) available from https://canvas.instructure.com/courses/801386/files/24062695]

On completion of this course, the student will be able to: 1. Describe the structure and operating characteristics of a high-performance microprocessor, and explain the principles of: orthogonal instruction set design; pipeline hazards and interlocks; branch prediction (both static and dynamic); out-of-order execution.

2. Explain the design and operating principles of arithmetic units including: high-speed adders and multipliers; dividers; and floating-point units. And also demonstrate how selected fixed-point arithmetic functions can be implemented (in a laboratory setting).

3. Design and implement both combinational and synchronous digital systems using state-of-the-art FPGA design tools and hardware description languages.

4. Describe the structure and operating characteristics of memory systems; demonstrate the ability to evaluate quantitatively the performance of a combined processor and memory system with respect to cycles-per-instruction (CPI) and memory bandwidth requirements; describe the operating principles of error detection and correction techniques applied to memory systems, and design a SECDED solution for a given memory system.

5. Reason about the ways in which memory hierarchies can be configured to exploit locality in order to reduce average memory access times, and quantitatively evaluate the impact of varying cache design parameters (e.g. capacity, associativity, block size, and write policies) on performance; understand the operating principles of cache coherency protocols, and be able to compare and contrast different implementation techniques.

Assessment Information

[Provide a description of all types of assessment that will be used in the course (e.g. written exam, oral presentation, essay, programming practical, etc) and how each of them will assess the intended learning outcomes listed above. Where coursework involves group work, it is important to remember that every student has to be assessed individually for their contribution to any jointly produced piece of work. Please include any minimum requirements for assessment components e.g. student must pass all individual pieces of assessment as well as course overall].

Types of assessment

This course will be summatively assessed using a combination of written exam and laboratory-based coursework exercises. The coursework reinforces lecture-based teaching by setting tasks in which students design and implement some aspect of a microprocessor, both in real hardware and also at higher levels of abstraction, using simulated systems. Therefore, a higher than normal weighting is given to the coursework (40%), with the remainder of the assessment (60%) coming from a written exam. This follows the structure currently used in the Computer Design course. There will be no requirement to pass any individual element of the coursework.

The quantity of computer architecture/design lab-based coursework that can be reasonably accommodated in a single-semester 20-point has been established over the past two years in the CD course. This course aims to include the same quantity, but now taking account of the fact that there is a wider set of learning outcomes to achieve.

The coursework will comprise three individually assessed exercises, scheduled and sequenced to align with the delivery of theoretical material in lectures. The first two exercises will be carried out in the computer design lab, and will involve:

a) An introduction to the Xilinx FPGA system, followed by simple exercises in designing, implementing and testing combinational and synchronous digital systems on FPGA using Verilog.

b) A more complex computer design exercise, which will typically involve designing some type of arithmetic circuit, and analysing its area and timing on silicon. We hope to have infrastructure in place for 2019 that will allow students to integrate their arithmetic design into an existing microprocessor, and thereby experience a very realistic design scenario (while keeping it all feasible for an undergraduate exercise).

The third assignment will be a simulation-based computer architecture exercise, based on the style of exercises previously set in the CAR course. This could be one of the following:

c) Design and implement a simulated branch predictor and evaluate its effectiveness on program traces supplied to the student.
d) Using a supplied cache simulator, explore the design space of cache hierarchies under a defined set of design constraints, with the aim of finding an optimum cache configuration.

The marks given, and time allocated, for each coursework assignment will be proportionate to the level of difficulty and expected effort involved in each assignment. The allocation of coursework marks will be 10%, 40%, and 50% for assignments 1, 2 and 3 respectively.

Relationship of assessments to learning outcomes

The learning outcomes are given in italics, with the associated method(s) of assessment explained thereafter.

1. *Describe the structure and operating characteristics of a high-performance microprocessor, and explain the principles of: orthogonal instruction set design; pipeline hazards and interlocks; branch prediction (both static and dynamic); out-of-order execution.*
   Most of these learning outcomes will be assessed primarily through the written exam. An understanding of branch prediction may be assessed by one of the coursework assignments.

1. *Explain the design and operating principles of arithmetic units including: high-speed adders and multipliers; dividers; and floating-point units. And also demonstrate how selected fixed-point arithmetic functions can be implemented (in a laboratory setting).*
   The computer design laboratory exercises will assess aspects of computer arithmetic that are feasible in a practical setting. For example, high-speed addition using parallel prefix adders, combinational multipliers, and sequential dividers, can be the subject of assessed laboratory design exercises. These cannot all be tested in one semester, due to time limitations, but instead exercises will be rotated from one year to the next.

2. *Design and implement both combinational and synchronous digital systems using state-of-the-art FPGA design tools and hardware description languages.*
   This is a generic design skill that will be assessed by both of the computer design laboratory exercises.

3. *Describe the structure and operating characteristics of memory systems,*
   This will be assessed by bookwork questions in the written exam paper.

   and evaluate quantitatively the performance of a combined processor and memory system with respect to cycles-per-instruction (CPI) and memory bandwidth requirements.
   This will be assessed through problem-solving exam questions, and sometimes in the third coursework assignment, when it is based on the cache simulation exercise.

4. *Reason about the ways in which memory hierarchies can be configured to exploit locality in order to reduce average memory access times, and quantitatively evaluate the impact of varying cache design parameters (e.g. capacity, associativity, block size, and write policies) on performance*.
These outcomes will be assessed either on the written exam paper on via the write-up for the third coursework assignment, when the cache simulation exercise is used.

describe the operating principles of error detection and correction techniques applied to memory systems, and design a SECDED solution for a given memory system.
These outcomes can both be assessed on the written exam paper, and the design aspect can also be examined in a laboratory exercise (this will not be possible for every cohort, due to the limited number of laboratory exercises we can set for a one semester, 20-point course).

Assessment Weightings:

Written Examination: 60%

Practical Examination: 0%

Coursework: 40%

Time spend on assignments:

[Weightings up to a 70/30 split between exam and coursework are considered standard, any higher coursework percentage requires a specific justification. The general expectation is that a 10-point course will have an 80/20 split and include the equivalent of one 20-hour coursework assignment (although this can be split into several smaller pieces of coursework. The Practical Examination category should be used for courses with programming exams. You should not expect that during term time a student will have more than 2-4 hours to spend on a single assignment for a course per week. Please note that it is possible, and in many cases desirable, to include formative assignments which are not formally assessed but submitted for feedback, often in combination with peer assessment.]

The breakdown of hours on this 20-point course includes a total of 45 hours of time spent on assessed coursework assignments. This justifies a higher weighting of coursework (40%).

Academic description:

[A more technical summary of the course aims and contents. May include terminology and technical content that might be more relevant to colleagues and administrators than to students.]

See the section on Academic Description provided on page 8.

Syllabus:

[Provide a more detailed description of the contents of the course, e.g. a list of bullet points roughly corresponding to the topics covered in each individual lecture/tutorial/coursework. The description should not exceed 500 words but should be detailed enough to allow a student to have a good idea of what material will be covered in the course. Please keep in mind that this needs to be flexible enough to allow for minor changes from year to year without requiring new course approval each time.]

The list of topics below is coloured to make it easier to identify those components that are derived from the current Computer Design course and those derived from Computer Architecture. Topics in black were previously covered (to some degree) in both courses. Topics in red were introduced in INF2C-CS and, in this course, are reviewed briefly and/or examined in greater technical depth (consistent with a level-10 course).

Each numbered item requires approximately one lecture in the schedule. It is possible, though not necessary, to interleave the Design and Architecture threads so that they run concurrently, to some extent.

1. Introduction; review of logic design, implementation technologies, combinational logic and logic minimization techniques.
2. The Verilog language.
3. Verilog in practice.
4. Sequential circuits and algorithmic state machines.
5. Register Transfer Level design principles; registers, clocks, timing budgets, setup and hold margins, clock skew, clock-domain crossings, synchronization, and metastability.
6. Quantitative computer architecture; performance evaluation methods and metrics, principles of high-performance design.
7. Instruction Set Architecture (ISA) design; instruction set classes, registers, memory addressing and organization.
8. RISC vs CISC, how the ISA supports high-level languages.
9. Quantitative approach to ISA design; example ISAs (e.g. MIPS, RISC-V); ISA requirements for embedded systems.
10. Review of pipelined processor design; pipeline hazards and interlocks; result forwarding.
11. Branch prediction techniques; static and dynamic schemes; two-level predictors, tournament predictors.
12. Out-of-order execution (I); scoreboards, reservation stations.
13. Out-of-order execution (II); register renaming, quantitative analysis of performance as a function of out-of-order architecture.
14. Introduction to binary arithmetic functions; fixed-point addition, subtraction, multiplication and division.
15. High-speed carry-lookahead adders and parallel-prefix adders.
16. Booth-coding for signed multiplication; Wallace and Dadda tree compressors; radix-4 Booth coding for tree-height compression.
17. Sub-word parallelism, fractional fixed-point multiply-accumulate operations.
18. Floating-point computations; IEEE standard, floating-point addition and multiplication, high-performance fused-multiply-add architectures.
19. Memory hierarchies; review of principles, quantitative analysis of memory hierarchy performance; exploring the design space of cache parameters (capacity, associativity, block size, write policy, allocation policy).
20. Cache coherency in multi-core architectures; e.g. MOESI protocol, snooping and directory-based coherency schemes.

21. Main memory design; structure of memory, interfacing between processor and memory, synchronous and asynchronous bus protocols; AMBA and AXI as examples of standard memory bus technologies.
22. Error detection and correction schemes; review of parity schemes, Hamming codes for error correction, SECDED.

Relevant QAA Computing Curriculum Sections:

[Please see http://www.qaa.ac.uk/en/Publications/Documents/SBS-Computing-consultation-15.pdf to check which section the course fits into.]

I111 – Computer architectures

I100 – Computer science

Graduate Attributes, Personal and Professional skills:

[This field should be used to describe the contribution made to the development of a student's personal and professional attributes and skills as a result of studying this course – i.e. the generic and transferable skills beyond the subject of study itself. Reference in particular should be made to SCQF learning characteristics at the correct level http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf].

Students completing this course can be expected to have developed the following personal and professional attributes and skills, beyond the study of the subject itself:

- The ability to exercise autonomy and initiative
- Advanced skills in the field of quantitative analysis of complex digital systems
- A working knowledge of how to optimize the performance of a system, using quantitative methods
- A specific understanding of the industrial design processes involved in microprocessor development, and a broad understanding of the ASIC and IP businesses as delivery vehicles for digital devices
- Practical skills in the design, implementation and testing of real-world digital designs, using a design flow of similar structure to that used today in industry
- The ability to demonstrate originality and creativity in dealing with professional level issues
- The ability to communicate effectively on a professional level with peers, senior colleagues and specialists

Reading List:

*[Provide a list of relevant readings. See also remarks under **3d**.]*

Textbooks

1. Hennessy/Patterson, Computer Architecture: A Quantitative Approach (5/e or 4/e).
2. Morris Mano, Michael D. Ciletti: Digital Design, 4/e, Prentice Hall, 2007.
3. Hamacher/Vranesic/Zaky: Computer Organization, McGraw-Hill, 2001.
4. Mano/Kime: Logic and Computer Design Fundamentals, Pearson, 2008.

5. Patterson/Hennessy: Computer Organization and Design, Elsevier, 2005.
6. Null/Lobur: The Essentials of Computer Organization and Architecture, Jones and Bartlett Publishers, 2003.

Students are strongly recommended to purchase either [1] or [5], and one from [2, 3, 4].

Breakdown of Learning and Teaching Activities:

[Total number of lecture hours and tutorial hours, with hours for coursework assignments.]

[The breakdown of learning and teaching activities should only include contact hours with the students; everything else should be accounted for in the Directed Learning and Independent Learning hours.

The total being 10 x course credits. Assume 10 weeks of lectures slots and 10 weeks of tutorials, though not all of these need to be filled with actual contact hours. As a guideline, if a 10-pt course has 20 lecture slots in principle, around 15 of these should be filled with examinable material; the rest should be used for guest lectures, revision sessions, introductions to assignments, etc.]

The hours expected for each type of learning and teaching activity is detailed in Table 1 below, for each relevant group of contiguous weeks during the semester. Activities involving contact hours are shown in bold.

Teaching takes place in three phases:

Phase 1 – *Introduction* (weeks 1 and 2)

- Phase 1 teaching is entirely lecture based, with **three lectures per week**
- Students have directed reading (10 hours per week)

Phase 2 – *Lectures and Lab-based assignments* (weeks 3 to 6)

- **Two lectures per week**
- **One 3-hour scheduled lab per week**
- Four hours prep/writeup outside of labs, covering the first two coursework assignments
- **One 1-hour feedback session per week**
- Students expected to spend 3 hours per week on reading and/or lecture prep

Phase 3 – *Lectures, tutorials and non-lab based assignment* (weeks 7 to 10)

- **Two lectures per week**
- **One tutorial per week**
- Final non-lab assignment expected to take 5 hours per week (20 hours total)
- Students expected to spend 3 hours per week on reading and/or lecture prep
- Two hours of assignment write-up per week

Deadline for completion of all coursework is the end of week 11, after which week 12 is set aside for revision, with the exam notionally in week 13.

This represents a total of 169 course hours, of which 38 are contact hours, with a total of 189 hours including the notional 20 hours of program-level activities.

Activity Type	Hours per week through the semester						TOTAL HOURS	CONTACT HOURS
	1 - 2	3 - 6	7 - 10	11	12	13		
Lectures	**3**	**2**	**2**				**22**	**22**
Scheduled lab time		**3**					**12**	**12**
Feedback sessions		**1**					**4**	**4**
Tutorials			**1**				**4**	**4**
Non-lab assignments			5				20	
Reading	10	3	3				44	
Lab prep/writeup		4	2	5			29	
Exam revision				8	13	9	30	
Exam						4	4	
Total course hours	26	52	52	13	13	13	169	
Course hours/week	13	13	13	13	13	13	13	
Program-level activities							20	
TOTAL							**189**	**42**

Table 1 - Detailed Learning and Teaching Activity Times

Lecture Hours: 22 hours

Seminar/Tutorial Hours: 4 hours

Supervised practical/Workshop/Studio hours: 12 hours

Summative assessment hours: 53 hours

Feedback/Feedforward hours: 4 hours

Directed Learning and Independent Learning hours: 74 hours

Program-level activities: 20 hours

Total hours: 189 hours

You may also find the guidance on 'Total Contact Teaching Hours' and 'Examination & Assessment Information' at:
http://www.studentsystems.ed.ac.uk/Staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

Keywords:

[A list of searchable keywords.]

Computer Design
Computer Architecture

Computer Systems

SECTION 3 - COURSE MATERIALS

3a. Sample exam question(s)

[Sample exam questions with model answers to the individual questions are required for new courses. A justification of the exam format should be provided where the suggested format non-standard. The online list of past exam papers gives an idea of what exam formats are most commonly used and which alternative formats have been http://www.inf.ed.ac.uk/teaching/exam_papers/.]

This course is the amalgamation of two well-established courses that have run for decades. Therefore, a large body of past exam papers exists, which are referred to here *in lieu* of sample exam questions. As this course represents a subset of the union of CD and CAR, the past exam papers from those two courses give complete coverage of the material in the syllabus of this course. We anticipate a similar format to the exam questions for this new course.

Computer Design Papers: see https://exampapers.ed.ac.uk/search/Computer+Design

Computer Architecture Papers: see https://exampapers.ed.ac.uk/search/Computer+Architecture

3b. Sample coursework specification

[Provide a description of a possible assignment with an estimate of effort against each sub-task and a description of marking criteria.]

Sample coursework specifications are provided via online documents that have been used in previous years for CD and CAR.

Computer Design Lab exercise 1:

- Assignment: https://www.inf.ed.ac.uk/teaching/courses/cd/PDF_2017/Handout_1.pdf
- Marking scheme: https://www.inf.ed.ac.uk/teaching/courses/cd/PDF_2017/Marking_1.pdf

Computer Design Lab exercise 2:

- Assignment: https://www.inf.ed.ac.uk/teaching/courses/cd/PDF_2017/Handout_2.pdf
- Marking scheme: https://www.inf.ed.ac.uk/teaching/courses/cd/PDF_2017/Marking_2.pdf

Computer Design Lab exercise 3:

- Assignment: https://www.inf.ed.ac.uk/teaching/courses/cd/PDF_2017/Handout_3.pdf
- Marking scheme: https://www.inf.ed.ac.uk/teaching/courses/cd/PDF_2017/Marking_3.pdf

Computer Architecture Assignment 1:

- Assignment: https://www.inf.ed.ac.uk/teaching/courses/car/Pracs/2017-18/Assignment1.pdf
- Marking scheme: see section 2, page 6, of the assignment document

Computer Architecture Assignment 2:

- Assignment: https://www.inf.ed.ac.uk/teaching/courses/car/Pracs/2017-18/Assignment2.pdf
- Marking scheme: see section 5, page 7, of the assignment document

3c. Sample tutorial/lab sheet questions

[Provide a list of tutorial questions and answers and/or samples of lab sheets.]

During Phase 1 there will be weekly example sheet handed out during lectures. Formative feedback will be available on these, or any other aspects of the course during the feedback sessions (or during scheduled lab sessions).

The four tutorials in Phase 2 will be based on the Computer Architecture material from 2017/18, and will be selected or adapted from the set of tutorial question/answer sheets below:

- Tutorial 1 Tutorial sheet 1 Solution 1
- Tutorial 2 Tutorial sheet 2 Solution 2
- Tutorial 3 Tutorial sheet 3 Solution 3
- Tutorial 4 Tutorial sheet 4 Solution 4
- Tutorial 5 Tutorial sheet 5 Solution 5

3d. Any other relevant materials

[Include anything else that is relevant, possibly in the form of links. If you do not want to specify a set of concrete readings for the official course descriptor, please list examples here.]

There are a number of external resource linked to on the previous year's CD web page. These are provided to enable students to gain a broader understanding of FPGA-based design and best-practices for designing digital circuits in general.

https://www.inf.ed.ac.uk/teaching/courses/cd/Resources.html

The slides from this year's Computer Design course are available from the LEARN website for Computer Design here:
https://www.learn.ed.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_64563_1&content_id=_3038431_1&mode=reset

This course will cover the material presented in slide sets 2-10, 14, and 15. This table of resources in LEARN also contains links to example sheets, as mentioned in 3c (this course is active, so the links go live according to a predefined schedule).

The slides from last year's Computer Architecture course are given on the web page below:

https://www.inf.ed.ac.uk/teaching/courses/car/slides.html

This course will incorporate the material from the first 7 of these slide-sets.

SECTION 4 - COURSE MANAGEMENT

4a. Course information and publicity

[Describe what information will be provided at the start of the academic year in which format, how and where the course will be advertised, what materials will be made available online and when they will be finalised. Please note that University and School policies require that all course information is available at the start of the academic year including all teaching materials and lecture slides.]

TBD

4b. Feedback

[Provide details on feedback arrangements for the course. This includes when and how course feedback is solicited from the class and responded to, what feedback will be provided on assessment (coursework and exams) within what timeframe, and what opportunities students will be given to respond to feedback.

The University is committed to a baseline of principles regarding feedback that we have to implement at every level, these are described at http://www.docs.sasg.ed.ac.uk/AcademicServices/Policies/Feedback_Standards_Guiding_Principles.pdf.

Further guidance is available from http://www.enhancingfeedback.ed.ac.uk/staff.html.]

This course builds on the experiences from CAR and CD, providing a range of mechanisms for feedback, both from staff to students and vice versa. **Feedback to students:** The previous CAR course ran weekly tutorials. During the tutorials, students would typically work through exam-like questions, and receive individual feedback. The previous CD course provided feedforward during the lab sessions, involving lecturer and demonstrators discussing work with class members, with the aim of improving their skills as they develop their coursework solutions prior to hand-in. There were also ad hoc feedback sessions, in which the lecturer gave feedback to students on their exercise sheet solutions, and worked through the solutions in an informal setting. Through SSLC feedback, these sessions were highlighted as useful, although students wanted more notice of when they would occur (which will happen in this new course, as all sessions are scheduled in advance). Both CD and CAR contained summatively assessed coursework, and in both cases written feedback was provided along with numerical grades. In this course, we propose a combination of mechanisms, including: feedforward during lab sessions, feedback on exercise sheet solutions during scheduled feedback sessions (in the first half of the course), and feedforward/feedback through tutorials (in the second half of the course). **Course feedback from students:** Both the CAR and CD courses have a track record of receiving positive feedback from students, via the University Course Enhancement Questionnaires (e.g. CD received 100% positive feedback

for the last two years). As this will be a new course, albeit based on two existing courses, our aim will be to solicit student feedback during the course, after each logical section of the syllabus has been covered, and after each assessed coursework item has been returned.

The UG3 SSLC also provides specific feedback, although this is usually too late in the semesters to have an immediate impact on the current year. Previous feedback from the SSLC has previously been incorporated into the delivery of the subsequent year.

4c. Management of teaching delivery

[Provide details on responsibilities of each course staff member, how the lecturer will recruit, train, and supervise other course staff, what forms of communication with the class will be used, how required equipment will be procured and maintained. Include information about what support will be required for this from other parties, e.g. colleagues or the Informatics Teaching Organisation.]

TBD

SECTION 5 - COMMENTS

[This section summarises comments received from relevant individuals prior to proposing the course. If you have not discussed this proposal with others please note this].

5a. Year Organiser Comments

[Year Organisers are responsible for maintaining the official Year Guides for every year of study, which, among other things, provide guidance on available course choices and specialist areas. The Year Organisers of all years for which the course will be offered should be consulted on the appropriateness and relevance on the course. Issues to consider here include balance of course offerings across semesters, subject areas, and credit levels, timetabling implications, fit into the administrative structures used in delivering that year.]

5b. BoS Academic Secretary

[Any proposal has to be checked by the Secretary of the Board of Studies prior to discussion at the actual Board meeting. This is a placeholder for their comments, mainly on the formal quality of the content provided above.]
