# ECE720T6: Trends in Reconfigurable Computing

Instructor: Nachiket Kapre (nachiket@uwaterloo.ca)

## Motivation

Reconfigurable architectures such as FPGAs (Field Programmable Gate Arrays) are enjoying a widespread renaissance in computing with big support from processor manufacturers like Intel [1], IBM [2] and even software vendors like Microsoft [3]. FPGAs are programmable hardware circuits that can be fully customized to the particular application thereby delivering accelerated performance, and energy efficiency. With its acquisition of Altera for $16.7B, Intel now ships computing chips dedicated for the cloud that combine a processor and an FPGA in the same chip. FPGA cards designed through the Microsoft Catapult project for Bing acceleration are being integrated into all machines in the massive Microsoft Azure data center. Students entering the technology workforce must be prepared to understand how FPGAs work, how they are programmed, and how to get the best use out of these platforms for their requirements.

[1] "Intel Bets $16.7 Billion on the Massively Parallel Future of Computing" — Motherboard.Vice 1st January 2016 — http://motherboard.vice.com/read/intel-bets-167-billion-on-the-massively-parallel-future

[2] "IBM bets POWER8 processor farm on hardware acceleration " — The Register 16th Nov 2015 — http://www.theregister.co.uk/2015/11/16/ibm_bets_power8_processor_farm_on_hardware_acceleration/

[3] "Microsoft Bets Its Future on a Reprogrammable Computer Chip" — Wired 25th September 2016 — https://www.wired.com/2016/09/microsoft-bets-future-chip-reprogram-fly/

## Content

This course will help you understand the foundational principles of FPGA architecture, the compilation and CAD process for modern FPGAs, high-level synthesis with OpenCL, communication overlay fabrics, and design strategies and patterns for computational acceleration. In particular, we will explore topics of contemporary interest such as machine learning that can influence both (1) how we design FPGAs, and (2) how they can be accelerated using FPGAs. This course will involve a combination of lectures, reading assignments, hands-on active labs, and a class project. We will use and study tools such as ABC, VTR (University of Toronto), IceStorm (Clifford Wolf/Vienna), InTime, and hardware platforms such as Intel HARPv2, IBM SuperVessel cloud, Microsoft Catapult, and development boards available locally at the University of Waterloo. The course will asses students with fortnightly assignment submissions and an

associated final class project (report+oral). There will be a final exam for this course which will be closely tied to the project execution. The fortnightly submissions will also be aligned with the class projects chosen by the students.

## Objectives

1. The organization of modern FPGA chips integrated heterogenous computing elements such as LUTs, FFs, DSPs, IO blocks, and even lightweight processors with rich connectivity between these elements through a programmable interconnect fabric. We will understand how this form of organization came to be and why this communication-rich design style represents a different, improved way to organize silicon resources than conventional CPUs. We will also briefly investigate how this template can be reused for other forms on computing such as quantum, and neural architectures.
2. FPGAs are typically programmed in low-level languages such as VHDL, or Verilog which are equivalent to painful assembly-level programming of CPUs. These descriptions are compiled through a series of stages such as synthesis, packing, placement, and routing. Each of these steps is an NP-complete problem and often takes hours to days of compile time for the largest FPGAs. We will study the underlying algorithms and study opportunities for acceleration. Specifically, we will learn about algorithms such as shortest path search, simulated annealing, set covering, and other graph-oriented computations. Faster CAD flows will help broaden the appeal of FPGAs to software developers used to the traditional rapid edit-compile-debug loops available with high-level languages. We will also discuss the idea of design space exploration and review tools and automation that can support this engineering process.
3. To overcome some of the challenges of describing hardware at low-level of abstraction using VHDL, and Verilog, many high-level approaches have been proposed using C/C++ and OpenCL. In this class, we will investigate (1) the use of OpenCL for Altera/Xilinx FPGAs and evaluate machine learning (or other) application mappings as a case study, and (2) understanding OpenCL specification for supporting soft processor overlays on top of FPGAs. This will involve learning about the OpenCL language specification and its applicability to the FPGA backend. We will assess the benefits and limitations of the OpenCL model for FPGA acceleration, and discuss ideas for overcoming them. If time permits, we will investigate the design of compiler passes for OpenCL optimizations in the pocl compiler framework.
4. Communication-rich FPGA architecture require the user design to expose all available connectivity in the circuit to the mapping tools. Instead, we can also use overlay NoCs (network-on-chip) on top of FPGA resources to support packet-switched and time-multiplexed communication between parts of the chip. We will study the different ways in which we can organize

these overlay networks, and associated challenges for design and workload mapping to these NoCs. This is directly tied to the adoption of FPGAs in data center systems such as the Intel HARP platform, or the Microsoft Catapult boards.

5. We will investigate the use of machine learning to enhance the FPGA CAD flow through better-quality predictive models, automated parameter tuning for vendor CAD tools, and for high-level compilers such as OpenCL. We will first learn about how to formulate machine learning problems and the range of algorithmic choices we have at our disposal. Then, we will apply these formulations to tackle CAD problems.

## Assessment Criteria

*Fortnightly* (20% credit): There will be regular homework assignments and reading tasks (once every two weeks). These assignments will be aligned with the class project chosen by the students.

*Mid-semester oral* (10% credit): This component is a short progress update that identifies the work that has been accomplished and work that still needs to be completed. This will be delivered in the form of 15-minute team presentation.

*End of semester report* (10% credit): Students will submit a written report (4 pages) based on their class project that should be in the form of a conference short paper. This will be in LaTeX with we will provide a template and build instructions to you.

*End of semester oral* (10% credit): Students will also do a final oral project presentation and a demo of their project. This will be delivered as a 20 minute presentation + 10 minutes Q&A session per team. A presentation template and examples of previous class presentations will be made available for reference.

*Final exam* (50% credit): Students will have a final exam for the course. The contents of the final exam will be closely tied to the class project.

## Details

**Workload**: 3 hours of lecture/week.

**Project**: The class projects must be chosen by the end of the second week of the class in teams of two. We will provide a list of pre-designed topics and expected outcomes to the student teams. Student teams may propose their own project which may be allowed at the discretion of the instructor.

## Learning Outcomes

At the end of the course, students should be able to: 1. Understand how modern FPGAs are organized, parse FPGA data sheets to extract key specifications, and identify suitable platforms based on application requirements. Extrapolate the technology to novel domains such as quantum, neural, or others. 2. Perform engineering tradeoffs in organization of the architectures, communication fabrics, to optimize for area, delay, or power metrics. Briefly reason about important criteria such as reliability, robustness, and fault tolerance. 3. Program, verify, and characterize CAD algorithms for FPGAs through (1) modification of the VTR or IceStorm codebase, or (2) bottom-up design of the algorithms. 4. Document the class project in the language of a technical paper, and deliver a high-quality group presentation.

## Pre-Requisites (not mandatory)

Students should have taken these courses. While not a strict requirement, it would be useful to have taken these courses.

**ECE606** - Algorithms — The CAD algorithms we need for this course will build upon those taught in ECE606. We will see how to adapt those algorithms to solve the FPGA implementation problem. **ECE621** — Computer Organization — this course is important to help contrast the FPGA organization style vs. Traditional CPU-based organizations. The more interesting FPGA devices are in fact hybrid FPGA-CPU chips.

## Co-Requisites (not mandatory)

Ideally, students should consider registering for the following concurrently offered course. With prior arrangement with the instructor, this requirement may be waived.

**ECE627** — RTL Digital Systems — Students should co-register for this Winter course as they will learn the key concepts of VHDL programming. This course will focus on OpenCL, but its important to understand how VHDL RTL design flow works.

## Syllabus

| Topic |
|---|
| Introduction to FPGAs |
| FPGA Tools Demo |
| FPGA Logic Organization |

| Topic |
| --- |
| FPGA Interconnect Design |
| CAD: Synthesis + Packing |
| CAD: Placement and Routing |
| HLS: Introduction to OpenCL |
| HLS: OpenCL Programming |
| Application Acceleration |
| Comm: FPGA NoCs |