CSE VLC Session 5
Flipped Classroom and Peer Instruction
Homework Assignment

Miad Faezipour

Course Title

CPEG 447 Logic Synthesis Using FPGAs

Brief Course Description: This course covers digital logic design using textual design entry and VHDL, behavioral, structural and data flow descriptions, technology-dependent vs. technology-independent design, logic synthesis and implied hardware for HDL codes, CPLD, rapid prototyping and retargeting designs for FPGAs. A major design project is given in this course.

Flipped Classroom

A Flipped Classroom is a pedagogical approach where students acquire knowledge before coming to class and engage in activities during class. Have you tried a flipped classroom? If so, please share the following in your homework assignment.

1) What do students do to prepare outside of class? Videos? Reading assignments? Homework exercises?
2) What do you have students do inside of class instead of lecture?
3) Provide 1-2 resources would you recommend for a professor considering this approach?

I have not tried flipped classroom for my course. However, since the course has a lab in addition to the lecture, students are involved in preparing material and gaining knowledge of certain design/codes prior to coming to the lab.
1) The students have reading assignments and homework exercises to prepare for the lab.
2) In the lab, students work together in groups of 2-3 on the FPGA boards and software tool to program the FPGA.
3) I have not applied flipped classroom for my course but would consider lecture slides, textbook and online resources for this approach.

Peer Instruction

Peer Instruction is a pedagogical approach where students spend much of the class time responding to multiple-choice questions and discussing answers in groups of 2-4.

In preparation for our next virtual meeting:

Read:

Simon et. al. 2010. Experience Report: Peer Instruction in Introductory Computing. *Proceedings of SIGCSE Symposium*.

Crouch & Mazur. Peer Instruction: Ten years of experience and results. *American Journal of Physics*. 69 (9), September 2001.

Watch:
There are many good videos posted by the Carl Wieman Science Education Initiative
http://www.cwsei.ubc.ca/resources/SEI_video.html

In particular:

- Student and Teachers Speak about Clickers
- How to use Clickers Effectively
- Anatomy of a Clicker Question

Write:
Create three clicker questions for your course that include appropriate distracters (wrong answers) and that are not too easy or too hard for students to do on their own.

I have not used clicker in my classroom, but it definitely seems to be a great way of having students actively involved in class and learning from their peers.

If I were to uses Clickers, the three questions below seem to be appropriate questions:

- The main difference between PLAs and FPGAs is:
  1) PLAs are programmable, FPGAs are not
  2) PLAs have AND/OR structure FPGAs do not
  3) PLAs and FPGAs are both programmable and have similar structures

- Consider the following Verilog Code:

```
module example (D, Clock, Q1, Q2);
   input D, Clock;
   output reg Q1, Q2;

   always @(posedge Clock)
      begin
       Q1 <= D;
       Q2 <= Q1;
      end
endmodule
```

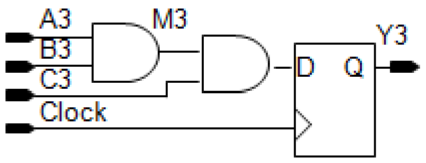
  1) The statements in the always block are Non-Blocking statements and would imply two cascaded flip flops.
  2) The statements in the always block are Blocking statements and would imply parallel /Concurrent flip flops.

- Consider the following VHDL code:

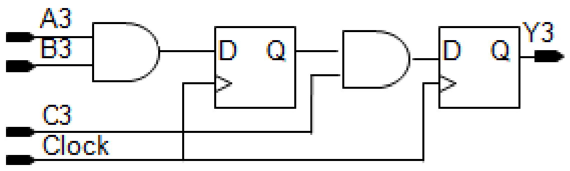
```
library IEEE;
      use IEEE.STD_Logic_1164.all, IEEE.Numeric_STD.all;

      entity SAMPLE_DESIGN is
            port (Clock, A1, B1, C1, A2, B2, C2, A3, B3, C3, A4, B4, 
C4: in std_logic;
                  Y1, Y2, Y3, Y4: out std_logic);
      end entity SAMPLE_DESIGN;

      architecture RTL of SAMPLE_DESIGN is
      begin

            PROCESS_ONE:
            process (A1, B1, C1)
                  variable M1: std_logic;
            begin
                  M1 := A1 and B1;
                  Y1 <= M1 or C1;
            end process PROCESS_ONE;


            PROCESS_TWO:
            process (A2, B2, C2, M2)
                  signal M2: std_logic;
            begin
                  M2 <= A2 and B2;
                  Y2 <= M2 and C2;
            end process PROCESS_TWO;


            PROCESS_THREE:
            process (Clock)
                  variable M3: std_logic;
            begin
                  if rising_edge(Clock) then
                        M3 := A3 and B3;
                        Y3 <= M3 and C3;
                  end if;
            end process PROCESS_THREE;


            PROCESS_FOUR:
            process (Clock)
                  signal M4: std_logic;
            begin
                  if rising_edge(Clock) then
                        M4 <= A4 and B4 after 3 ns;
                        Y4 <= M4 and C4 after 1ns;
                  end if;
            end process PROCESS_FOUR;
      end architecture RTL;
```

The implied hardware for PROCESS_THREE looks like:

1)



2)



3) None of the above.