Electronics Systems Engineering
Faculty of Engineering and Applied Science, University of Regina

ENEL 489/ ENEL 789 Lab 2
IP cores and FPGA implementation

Purpose:

1. How to use the IP cores
2. Implementation of project on Zedboard Zynq evaluation kit

Equipment:

Lab computers in ED434, Xilinx Vivado 2016.3, Zedboard Zynq evaluation kit

Fixed Point Data Representation:
Mathematical operations using floating point representation are very time consuming and costly. The alternative to using floating point representation is fixed point representation. As the name suggests, in fixed point representation, the decimal place remains fixed. Fixed point arithmetic is essentially just integer arithmetic, and as such is much faster, and easier to implement. Details of fixed-point data representation will be examined in the lab. We will be using the fixed-point format of Q8.8 for the implementation of this lab.

Procedure:

Start with Task 3.

Task 1.a: design a Q8.8 fixed point multiplier using * operator

1. Create a new project in Xilinx Vivado as discussed in previous labs
2. Define Module window should come up. This window allows a user to enter in the external port declarations. Enter the following.
   Note: MSB and LSB are for buses only.
   - CLK – in
   - A – in (MSB 15 – LSB 0)
   - B – in (MSB 15 – LSB 0)
   - P – out (MSB 15 – LSB 0)
3. Write the VHDL code to implement a Q8.8 fixed point multiplier using * operator.
4. Write the testbench and verify the operation of module.

Task 1.b: design a Q8.8 fixed point multiplier using IP core

1. Create a new project in Xilinx Vivado as discussed in previous labs
2. Define Module window should come up. This window allows a user to enter in the external port declarations. Enter the following.
   Note: MSB and LSB are for buses only.
   - CLK – in
   - A – in (MSB 15 – LSB 0)

ENEL 489/ ENEL 789 Lab 2
IP cores and FPGA implementation

- B – in (MSB 15 – LSB 0)
- P – out (MSB 15 – LSB 0)

3. In **Flow Navigator** window, click on **IP Catalog**. It will open a new window titled **IP Catalog**. Navigate to **Vivado repository>Math functions>Multipliers>** and double click on **Multiplier**. A new window will be opened in which different options of multipliers can be used. From the **Basic** tab set data type to be signed and width to be 16 bits for both the operands. Explore the **Multiplier Construction** and **Optimization options** and their effect on **Resource estimates** available under **Information** tab on the left side of window. Configure to **use mult** and **optimize for speed.** Under the **Output and Control** tab, check use custom output width and select the appropriate output 16 bits in accordance with Q8.8 format. Give it an appropriate component name e.g. **mult_ip** and click okay.
4. The multiplier is added in the project and you can use it as a "component" in your module defined in step 2. Just remember, you still need to add component definition in your own module as shown below. Instantiate the IP multiplier, bind its port with your module appropriately

```
component mult_ip IS   PORT (
                    CLK : IN STD_LOGIC;
                    A : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
                    B : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
                    P : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
                  );
 END component;
```

5. Use the testbench of task 1.a to verify the operation of this multiplier.

Task 2.a: design a Q8.8 fixed point adder using + operator

1. Design a Q8.8 adder using + operator and verify its functionality via testbench

Task 2.b: design a Q8.8 fixed point adder using IP core

2. Locate Adder/Subtractor IP core and use it to design a Q8.8 adder. Verify its functionality via testbench

Task 3: Design a simple 4-bit adder and implement it on Zedboard Zynq kit.

The purpose of this task is to get familiarity with the process of implementation on Zedboard Zynq kit.

1. Create a new project and design a simple clock enabled 4-bit adder.
2. Add the provided "lab3.xdc" constraint file and modify it to utilize the 8 sliding switches SW0 to SW7 as two 4-bit inputs and 4 LEDs LD0 to LD3 as outputs. This constraint file will be discussed during lab session.
3. Synthesize the project and implement it.

ENEL 489/ ENEL 789 Lab 2
IP cores and FPGA implementation

4. After implementation, Generate the bitstream.
5. Connect the Zynq kit with your computer and click Open target under hardware manager. Select the kit and download the bitstream.
6. Use the slide switches and LEDs to verify the functionality.

Instructor

Syed Aamir Ali Shah

Email: aamirshah@uregina.ca

Grading:

- Labs are due at the beginning of your next lab and will be graded during the lab period.
- You will be verifying the functionality of each task by running the simulations in order to get your labs graded.
- There is no formal lab report. However, you may be asked to turn in diagrams or other design documents, or answer questions.