CPEG 422 Project 3

IP Packaging, Block Design and Processor-FPGA communication

During this project, you will learn to pack your IP, implement block design, and observe communication between the microprocessor and the FPGA. You are required to implement a Multiply-Accumulate (MAC) unit IP in two different levels: high-level and gate-level. You are supposed to first pack your own MAC IP, then construct block design based on your IP, and finally test and observe the communication between the processor and the FPGA. At the end of this project, you need to demo your systems and submit report.

Goals of this project:

Learn to create IP
Learn how to use Zybo board
Learn how to coordinate the processor and the FPGA

Your tasks:

- Create and pack your MAC IP in high-level and gate-level
- Build a MAC based block design and generate bistreams for FPGA
- Program the FPGA and the processor, coordinate the communication between processor and FPGA
- Write project report

Software/Hardware and Toolkit:

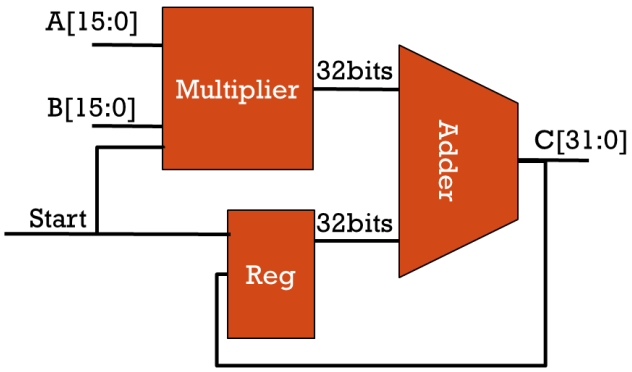
- Vivado 2017.4 (For creating IP and building block design)
- Zybo (For implementing block design and implementing communication system)
- Xillinx SDK (For software control on Processor side)
- Serial terminal (For observing communication between Processor and FPGA)

MAC IP:

A MAC unit performs multiplication and accumulation process. *Note that all the computation should be in **signed 2's complement** form.* A basic MAC unit consists of a multiplier and an adder. It a basic unit in Digital Signal Processing (DSP).

- **Block diagram**

  A [15:0]: 16-bit multiplier
  B [15:0]: 16-bit multiplicand
  C [31:0]: 32-bit accumulated results
  Start: a control signal. Each time start enables, **c=a*b+c** is executed. The register (initially 0) captures the current accumulation result for next round of computation.



- **High level design:**

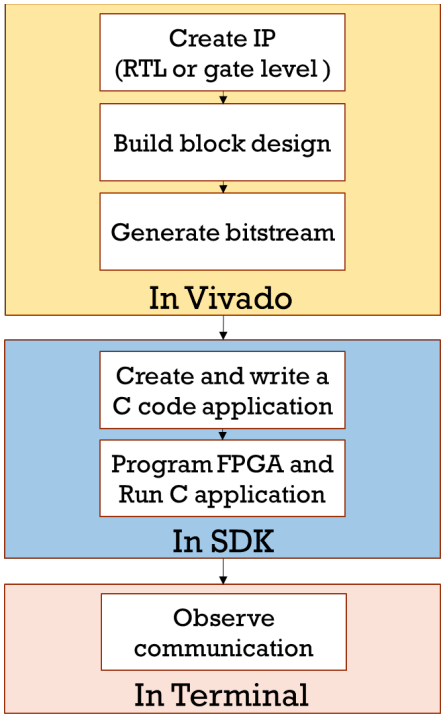
Use high level behavior description to implement the MAC function in IP. Such as:
c=a*b+c;

- **Gate level design:**
  You will create a gate level MAC design composed of a multiplier and an adder designed from previous projects. Use the given modules to implement the MAC function in your IP. **No '+' or '*' is allowed.** The top-level module should be used as a component in the IP.

Project flow:

The figure shows the major steps of the project. Three tools are used in order:

1. **In Vivado**:
   1) Create your MAC IP: you should use AXI4 Peripheral IP template as the starting point of your IP, add your MAC logic in the template (both high-level method and gate-level method).
   2) Build block design: create a block design, add your **IP** and **ZYBQ** processor system in design. Then make automatic connection.
   3) Generate bitstream: Click generate bitstream, once it successfully finishes synthesis and implementation, the bitstream is ready.
2. **In SDK**:
   1) Build a software application: create a C program, use the program to control the FPGA by setting inputs, sending control signals, and receiving outputs from FPGA.
   2) Click Program FPGA and Execute the C program.
3. **In Terminal**: Observe the processor-FPGA communication on the screen.



Typically, you need to run the entire flow twice, one time for RTL high-level IP and one time for gate-level IP.

Report requirements:

1. Design summary:
   1) Summarize how your implement your IP with gate-level code;
   2) Compare gate-level and high-level IP implementations, describe the changes made in the IP code and the changes made in C program for the gate-level implementation.
2. Result display:
   1) Briefly describe how C program controls and communicates with FPGA design.
   2) Show your C code for both high-level level IP and gate-level IP.
   3) Snapshot the communication displayed on terminal for high-level level IP and gate-level IP.

3. Design implementation report: Summarize the block design implementation report offered in Vivado. Please report the following for high-level level IP and gate-level IP:
    1) Hardware utilization of post-implementation (FF,IO ,LUT, BUF…) in table format.
    2) Power report (dynamic vs. static, also signal, logic and I/O).
    3) Timing report (critical path delay).
    4) Design schematic snapshots.
    5) Compare high-level and gate-level designs, summarize the differences.

Due Dates:

Project Demo:	04/29 in class
Report Submission:	05/01 at midnight